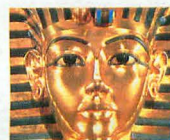
1989年11月1日発行(毎月1回1日発行)第8巻11号通巻91号　昭和58年11月2日第三種郵便物認可
ISSN 0910-7614
パーソナルコンピュータ・マガジン
MZシリーズ,X1/turbo,X68000&ポケコン
Oh!X
オー!エックス　定価560円
特集
micro Computer入門
8ビットRISC(?)プロセッサもどきの設計と製作
高性能32ビットマイクロプロセッサの話
X68000&X1周辺LSIの使い方
マシン語カクテル in Z80's Bar
X1で3重スクロールゲームを作る
X68000
X-BASIC調理実習/マシン語プログラミング
C調言語講座PRO-68K/DōGA・CGA講座
カードゲームばばぬき
S-OS
TTI用パズルゲームPUSH BON!
THE SOFTOUCH
リングマスター1 フィリアス・ノギスの暗雲
Stationery PRO-68K
LIVE in '89
X1/turboオブ・ラ・ディ,オブ・ラ・ダ
X68000メタル・ホーク
猫とコンピュータ/知能機械概論
MZ-2500グラフィックエディタ作成講座
(で)のショートプロぱーてい
11
NOV. 1989

夢のつづきを語ろう。

「少なくとも、同じベクトルをもつスタッフで仕事をしたい」、クリエイターの間でよく言われることですが、黙っていても意志の通じ合う、そんな環境がさらにいい仕事を喚起してくれるものです。X68000を囲むユーザー、ソフトハウス、パブリッシャー、ハードベンダー、そして私たちメーカーの関係も、まさにそうした絆を感じさせるものがあるといえば、奢りでしょうか。これまで着実に培われてきた、そしていま目の当たりにするX68000の環境にも、同じ“のり”のもとでますます活性化する使用環境、さらに新たな指標をめざすパワフルなトレンドが息づいています。プロの技法をサポートした上でヒューマンインターフェイスをも追求した高感度アプリケーション。また多彩なペリフェラルのサポートで、さらに高次元な領域へと踏み込めるシステム環境。先鋭なアーティスティックな側面と、ホリゾンタルなマシンとしての不偏性。潜在能力がまたひときわ光彩を放ちます。

〈共通特長〉●さらに高い次元へと進化した処理機能とヒューマンインターフェイス、 Human68k ver2.0、日本語フロントエンドプロセッサver2.0搭載●プロセッサの未来を先取した68000搭載●テキスト、グラフィック、スプライトの3画面を独立させた独自のメモリアーキテクチャー●1024×1024ドット(最大表示エリア768×512ドット)、高品位な金属までも自然に表現しうる65,536色同時発色(512×512ドット時)の高解像度自然色グラフィックス●16×16ドットの緻密なキャラクタを駆使できるスプライト機能(水平32スプライト、1画面128スプライト、65,536色中16色)●リアルなサウンドシーンをクリエイトできるステレオFM音源に加え、サンプリング音源としてAD PCM搭載●オートロード、オートイジェクトメカ採用、インテリジェントな1Mバイトの5"FDD2基搭載●蓄積された多彩なジャンルのアプリケーションが利用できるX68000シリーズとソフトコンパチ。

〈EXPERTシリーズ〉高密度実装を象徴するフォルム、マンハッタンシェイプ●新たな領域をひらく3Mバイトの大容量メモリを標準装備、メインメモリは標準で2Mバイト、最大12Mバイトまで拡張可能●40Mバイトハードディスク搭載(CZ-612C)※●マウス・トラックボール標準装備●日本語入力にスムーズに対応するASCII準拠フルキーボードを採用。

〈PROシリーズ〉●意表をつくボディコンストラクション、高度な実装技術に裏付けられた洗練と信頼性の、新しいスタンダードフォルム●高度なシステム化への対応を考慮した4スロットの拡張I/0スロット標準装備●プロニーズに対応した大容量ファイル、40Mバイトハードディスク搭載(CZ-662C)※●2Mバイトの大容量メモリを標準装備●マウス標準装備●使いやすいワイドスケールのフルキーボード。

※CZ-602C、CZ-652Cには、本体内に内蔵できる増設用の40Mバイトハードディスクドライブ(CZ-64H標準価格120,000円税別・取付費別)をサポート。

●写真左はCZ-612C-BK＋CZ-612D-BK、写真右はCZ-652C-GY＋CZ-603D-GY

X68000見体験フェア	●11月より全国各地で開催します。
EXEリーダーズ「カップ」プレゼント実施中	●いま、EXE会員よりご紹介のお客様がEXEショップでX68000シリーズを購入されますと、EXE会員にEXEリーダーズ「カップ」をプレゼントします。詳しくはEXEショップにお問い合わせください。 ●また、X68000シリーズをご購入のお客様は、ぜひEXEクラブにご入会ください。

本広告に掲載しております商品および役務の価格には消費税は含まれておりませんので、ご購入の際、消費税額をお支払い下さい。

シャープ株式会社
●お問い合わせは…シャープ(株)電子機器事業本部システム機器営業部〒545大阪市阿倍野区長池町22番22号☎(06)621-1221(大代表)
電子機器事業本部テレビ事業部第4商品企画部〒162東京都新宿区市谷八幡町8番地☎(03)260-1161(大代表)

表紙絵： Moto Noriyuki

■広告目次

Oh!X

C O N T

●特集

●カラー紹介

●読みもの

〈スタッフ〉
●編集長／前田 徹 ●副編集長／永野 仁 ●編集／植木章夫 石塚康世 太田慎一 岡崎栄子 ●協力／有田隆也 中森 章 清水和人 後藤貴行 林 一樹 荻窪 圭 岡本浩一郎 毛内俊行 吉田賢司 影山裕昭 相馬英智 古村 聡 村田敏幸 丹 明彦 三沢和彦 長沢淳博 宮島 靖 金子俊一 ●カメラ／杉山和美 ●イラスト／永沢しげる 山田晴久 小栗由香 ●アートディレクター／島村勝頼 ●レイアウト／元木昌子 AD GREEN ●校正／千野延明 織田洋子

1989 NOV.
11

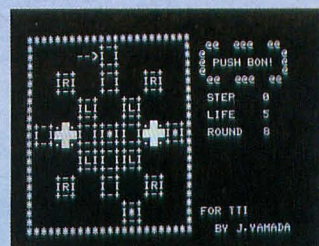
E N T S

●THE SOFTOUCH

●シリーズ全機種共通システム

●連載/紹介/講座/プログラム

特集micro Computer入門

マシン語カクテル in Z80's Bar

PUSH BON!

カードゲームばばぬき

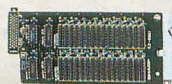
DōGA・CGA

リングマスターI

X68000をサポート。

シャープペリフェラルファミリー
X68000

ボード

拡張メモリ

1MB増設RAMボード
(CZ-600C用)
CZ-6BE1
標準価格 35,000円(税別)

1MB増設RAMボード※3
(CZ-601C/611C/652C/662C用)
CZ-6BE1A
標準価格 38,000円(税別)

2MB増設RAMボード※4
CZ-6BE2
標準価格 79,800円(税別)

4MB増設RAMボード※4
CZ-6BE4
標準価格 138,000円(税別)

インターフェイス

ユニバーサルI/Oボード
CZ-6BU1
標準価格 39,800円(税別)

GP-IBボード
CZ-6BG1
標準価格 59,800円(税別)

増設用RS-232Cボード
(2チャンネル)
CZ-6BF1
標準価格 49,800円(税別)

数値演算プロセッサ

数値演算プロセッサボード
CZ-6BP1
標準価格 79,800円(税別)

FAX

FAXボード
CZ-6BC1
標準価格 79,800円(税別)

MIDI

MIDIボード
CZ-6BM1
標準価格 26,800円(税別)

ネットワーク

モデム

モデムユニット※5
CZ-8TM2
標準価格 49,800円(税別)
(RS-232Cケーブル同梱)

RS-232Cケーブル

RS-232Cケーブル
(平行接続型)
CZ-8LM1
標準価格 7,200円(税別)

RS-232Cケーブル
(クロス接続型)
CZ-8LM2
標準価格 7,200円(税別)

入力

インテリジェントコントローラ NEW
CZ-8NJ2
標準価格 23,800円(税別)

マウス・トラックボール NEW
CZ-8NM3
標準価格 9,800円(税別)

トラックボール
CZ-8NT1
標準価格 13,800円(税別)

マウス
CZ-8NM2A
標準価格 6,800円(税別)

ジョイカード
CZ-8NJ1
標準価格 1,700円(税別)

その他

拡張スロット

拡張I/Oボックス(4スロット)
(CZ-600C/601C/611C/602C/612C用)
CZ-6EB1
CZ-6EB1-BK
標準価格 88,000円(税別)

スピーカー

アンプ内蔵
スピーカーシステム(2本1組)
AN-S100
標準価格 36,600円(税別)

システムラック

システムラック
CZ-6SD1
標準価格 44,800円(税別)

※4 ご使用に際しては、あらかじめ別売の1MB増設RAMボードCZ-6BE1 標準価格35,000円(税別・CZ-600C用)、CZ-6BE1A 標準価格38,000円(税別・CZ-601C、CZ-611C、652C、662C用)を増設してください。
※5 モデムユニットCZ-8TM2に同梱のソフトはX1/X1ターボシリーズ用です。

●ミニフロッピーディスクユニット(2D)	★CZ-502F	99,800円
●ミニフロッピーディスクユニット(2D・1ドライブ)	CZ-503F	49,800円
●増設用ミニフロッピーディスクドライブ(2D)※4	CZ-53F-BK	19,800円

拡張ボード・その他

●モデムユニット(300/1200ボー)	CZ-8TM2	49,800円
●320KB外部メモリ	CZ-8BE2	29,800円
●RS-232C・マウスボード※5	CZ-8BM2	19,800円
●フロッピーディスクインターフェイス※6	CZ-8BF1	14,800円
●JIS第1水準漢字ROM※7	CZ-8BK2	19,800円
●RS-232C用ケーブル(平行接続型)	CZ-8LM1	7,200円
●RS-232C用ケーブル(クロス接続型)	CZ-8LM2	7,200円
●拡張I/Oボックス	CZ-8EB3	33,800円
●RFコンバータ※8	AN-58C	2,980円
●インテリジェントコントローラ	CZ-8NJ2	23,800円
●マウス・トラックボール	CZ-8NM3	9,800円
●マウス	CZ-8NM2A	6,800円
●トラックボール	CZ-8NT1	13,800円
●ジョイカード	CZ-8NJ1	1,700円
●チルトスタンド※9	CZ-6ST1-E・-B	5,800円
●高性能CRTフィルター※10	BF-68PRO	19,800円
●スキャナ用パラレルボード※11	CZ-8BN1	27,800円

●品番中の-表示は、B<ブラック>・E<オフィスグレー>を示します。※1 X1ターボZシリーズ用 ※2 CZ-862Cには接続できません ※3 X1ターボシリーズ用 ※4 CZ-830C用 ※5 X1シリーズ用 ※6 CZ-850CでCZ-520Fを使用する場合に必要 ※7 CZ-800C、801C、802C、803C、811C、820C用 ※8 CZ-820C、822C、830C用 ※9 CZ-600D、880D、830D用 ※10 14/15型用 ※11 CZ-8NS1用 ●接続等の説明につきましては、周辺機器総合カタログをご参照ください。

★印の商品は在庫僅少です。

本広告に掲載しております商品および役務の価格には消費税は含まれておりませんので、ご購入の際、消費税額をお支払い下さい。

電子機器事業本部テレビ事業部第4商品企画部 〒162 東京都新宿区市谷八幡町8番地 ☎(03)260-1161(大代表)

EXE会員の集い 見・体・験フェア開催!!

イベント1
X68000相談コーナー
★X68000の事なら何でもお答えします。

イベント2
オリジナル作品発表会
★君の作品を仲間に伝えよう! 君の苦労を仲間と語ろう!

イベント3
ミニEXEスクール
★新作ソフト勉強コーナー、X68000の魅力をじっくりと学ぼう!

イベント4
新作ゲーム大会
★君はチャンピオンになれるか!!(豪華景品多数用意)

ご来場記念品進呈!
お友達と一緒に
X68000ファン全員集合!!

EXE会員の集い

開催地区	開催日	会場	主催・問い合せ先
北海道	11/5(日)	札幌京王プラザホテル・2Fローズルーム	シャープエレクトロニクス販売㈱北海道統轄営業部 ☎011-642-8111㈹
北関東	12/2(土)、3(日)	水戸市民会館	シャープエレクトロニクス販売㈱北関東統轄営業部 ☎0286-35-1151㈹
首都圏	11/12(日)※	東京・外神田大同毛織ビル9F大同ホール	東京中央シャープ販売㈱ ☎ 03-833-1611㈹
	12/2(土)、3(日)	東京・新宿エルタワービル・イベントホール	シャープエレクトロニクス販売㈱首都圏統轄営業部 ☎ 03-266-8248
中部	11/25(土)、26(日)	シャープ名古屋ビル・7Fホール	シャープエレクトロニクス販売㈱中部統轄営業部 ☎052-332-2611㈹
中国	11/4(土)、5(日)	ソシミエール岡山玉姫殿・3F鷲羽の間	シャープエレクトロニクス販売㈱中国統轄営業部 ☎082-874-2282㈹
九州	12/9(土)	博多シティホテル・5F高千穂の間	シャープエレクトロニクス販売㈱九州統轄営業部 ☎092-501-6806

主催:シャープエレクトロニクス販売㈱、東京中央シャープ販売㈱/後援:シャープ㈱、〔X68000EXEショップ、X68000ユーザーズクラブ〕

※11/12(日)東京は招待客のみ。

HARD DISK UNIT

SHARP X68000 専用

IT X640/680

初めまして。私、専用機です。

充実した新機能

1. ハードディスク内蔵タイプにも接続することができます。
2. ID番号切り替えロータリースイッチにより1台〜最大8台まで接続が可能です。(Human 68K Ver2.0以上使用の事)
3. ニューデザインで色もX68000シリーズ本体に合わせてブラック・グレーの2色を用意しました。
4. OS-9対応。
5. 従来の当社製X68000シリーズ対応のHDU (ITX-403/203等)にも増設が可能です。(ただし従来の機種は1台目は固定)

※IT X640はMSX2 HD Interfaceにも接続することが可能。

■IT X640……………………………¥158,000
- 40MB HDU
- 平均アクセスタイム28ms

■IT X680……………………………¥198,000
- 80MB HDU
- 平均アクセスタイム20ms

※40MB×2の設定されています。

体験してください。

データショウ'89に出展!
日時:1989年10月24日(火)〜10月27日(金)
東京晴美展示会場　南館1F ブースNo.44

※IT X640/680とも接続の際ターミネータ別売¥4,000が必要です。(従来機種の場合は必要ありません)

アイテック株式会社

本社 〒550 大阪市西区新町1丁目3番12号 TEL:(06) 532-0216 FAX:(06) 532-7253
関西営業所 〒550 大阪市西区新町1丁目25番14号 TEL:(06) 532-0120 FAX:(06) 532-0543
関東営業所 〒275 千葉県習志野市谷津1丁目12番5号 TEL:(0474)77-7564 FAX:(0474)73-2759
名古屋営業所 〒460 名古屋市中区大須2丁目28番31号 TEL:(052)212-1487 FAX:(052)212-1627

インフォメーションセンター TEL:(06)532-0320

プロのための3次元コンピューター グラフィックス

C-TRACE

◆建築シュミレーション

◆究極のグラフィック

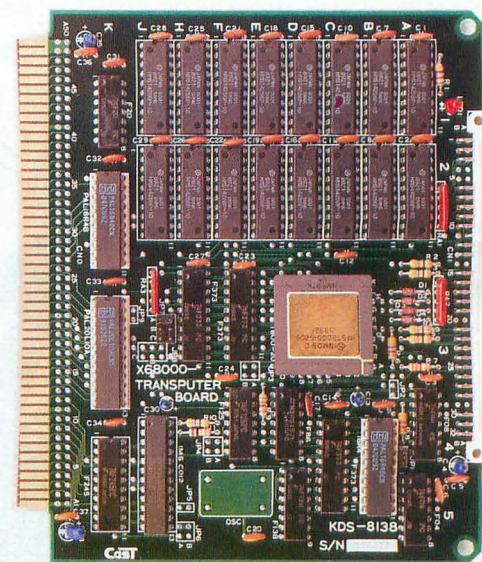
◆自由な質感、
自由なアングル

SPEED UP-MACHINE
超高速レイトレーシング

TRANS PUTER

新発売!!

※本商品は店頭販売いたしません。
直接当社までお申し込みください。

◆スタジオ、モデル不要

C-TRACE 68TP■C-TRACE 98TP
(ソフト・ハードセット) ¥610,000

C-TRACE TOWNS	¥68,000
C-TRACE 68 (X68000対応)	¥68,000
C-TRACE 98 DRY (PC-9801対応)	¥68,000
C-TRACE 98+ (PC-9801対応)	¥198,000
C-TRACE NEWS (SONY)	¥380,000

ディスプレイのマッハバンド(しま模様)が気になる方へ。
きれいなビデオ出力が欲しい方へ。1670万色同時表示、
C-FRAME68 新発売!!
フルカラー・フレームバァッファ、コンポジット入出力機能内蔵。
ペイントソフト付き ¥248,000
もちろん、C-TRACE68も対応。

▶これだけあれば後はいらない!?
CG98フルコース ¥138,000(PC-9801全機種対応)
フレーム・バッファ, ペイント, ポリゴン, レイトレ のフルセット
このセットでCGのすべての分野が体験できます。
※この製品は直接当社までお申し込みください。

株式会社キャスト 〒158 東京都世田谷区等々力2-1-13 TEL.03-705-0656 FAX.03-705-5224 ★男女従業員募集中!

68ユーザーに夢の高速エディタを――。その名もジェイムス68K!!

James 68k――――X68000のハードウェア性能を最大限に利用した高速マルチスクリーン・エディタ。X68000ユーザーの方に絶対の自信をもって、お推めできるNICE GUYです。

特長1

ウィンドウ切替およびスクロールはビットマップ方式のテキスト画面でありながら、テキストマップ方式に負けないスピードを実現。

特長2

ウィンドウは最大50/さらに内容を確認しながらウィンドウを自由に切り替えることが可能。

特長3

マークジャンプ位置をワンタッチで設定、高速ジャンプすることができ、ウィンドウ間での自由な往来が可能。

特長4

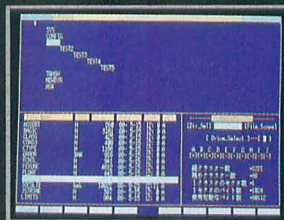
マクロを最大4つまで定義・実行することが可能。

特長5

ディスク階層ディレクトリの高速表示、ファイル内容の一部表示など新機能を搭載し、編集ファイルの選択が簡単。

好評発売中

高速日本語マルチスクリーン・エディタ

JAMES 68K

●動作環境

本体	X68000シリーズ
OS	Human 68K V1.0/2.0
CRT	X68000専用ディスプレイ
その他	ハードディスク対応
定価	¥20,000(税別)

開発・発売元

YET ワイ・イー・ティ/〒720 広島県福山市今町ブルートレイン内 TEL.(0849)22-2411

※通信販売ご希望の方は、商品名、機種名、住所、氏名、電話番号を明記の上、現金書留にてお申し込み下さい。(送料サービス)

超低金利クレジットOK!!

またまた秋葉原で おなじみの

注目!!
冬のボーナス一括払いOK!!
手数料(金利)無料
(12月末払い、ご利用下さい)

●お近くの方はお
●本体単品で特
●ビジネスソフト定

10/15〜11/20

CYBER STICK
●CZ-8NJ2
(定価¥23,800)
超特価!!
▶価格はTEL下さい

X-1ターボZⅢ 特別ご提供品!! 台数限定
●CZ-888C+CZ-860D+M-2HD(10枚)
定価¥269,600▶特価¥164,800
・ジョイカード
・ゲーム3種
・パソコンラックⒶ3段
プレゼント中
(ボーナス併用も有りますTEL下さい)
送料消費税込み!!

12回	14,300	24回	7,500	36回	5,100	48回	4,000	60回	3,300

ジョイスティック 送料¥500
●X-1PRO
定価¥9,500▶特価¥7,800
●ASCII STICK
定価¥6,800▶特価¥5,500

P&A超低金利クレジットをご利用ください!!

X68000EXPERT & EXPERT-HD (送料消費税込み)

EXPERT
(ボーナス併用も有ります。TEL下さい)

Ⓐセット:CZ-602C+CZ-603D……定価¥440,800▶現金価格はお電話下さい

12回	29,100	24回	15,200	36回	10,500	48回	8,100	60回	6,800

Ⓑセット:CZ-602C+CZ-602D……定価¥455,800▶特価(現金価格はお電話下さい)

12回	30,700	24回	16,100	36回	11,000	48回	8,600	60回	7,100

Ⓒセット:CZ-602C+CZ-612D……定価¥475,800▶特価(現金価格はお電話下さい)

12回	32,000	24回	16,700	36回	11,500	48回	8,900	60回	7,400

Ⓓセット:CZ-602C+CU-21CD……定価¥495,800▶特価(現金価格はお電話下さい)

12回	32,500	24回	17,000	36回	11,700	48回	9,100	60回	7,600

EXPERT-HD

Ⓐセット:CZ-612C+CZ-603D……定価¥550,800▶特価(現金価格はお電話下さい)

12回	35,900	24回	18,800	36回	12,900	48回	10,000	60回	8,400

Ⓑセット:CZ-612C+CZ-602D……定価¥565,800▶特価(現金価格はお電話下さい)

12回	37,800	24回	19,800	36回	13,600	48回	10,600	60回	8,800

Ⓒセット:CZ-612C+CZ-612D……定価¥585,800▶特価(現金価格はお電話下さい)

12回	38,700	24回	20,300	36回	13,900	48回	10,800	60回	9,000

Ⓓセット:CZ-612C+CU-21CD……定価¥605,800▶特価(現金価格はお電話下さい)

12回	39,300	24回	20,600	36回	14,100	48回	11,000	60回	9,200

X68000PRO & PRO-HD (送料消費税込み)

PRO
(ボーナス併用も有ります。TEL下さい。)

Ⓐセット:CZ-652C+CZ-603D……定価¥382,800▶特価(現金価格はお電話下さい)

12回	25,200	24回	13,200	36回	9,100	48回	7,000	60回	5,900

Ⓑセット:CZ-652C+CZ-602D……定価¥397,800▶特価(現金価格はお電話下さい)

12回	26,800	24回	14,000	48回	9,600	48回	7,500	60回	6,300

Ⓒセット:CZ-652C+CZ-612D……定価¥417,800▶特価(現金価格はお電話下さい)

12回	28,200	24回	14,700	36回	10,100	48回	7,900	60回	6,600

Ⓓセット:CZ-652C+CU-21CD……定価¥437,800▶特価(現金価格はお電話下さい)

12回	28,500	24回	15,000	36回	10,300	48回	8,000	60回	6,700

PRO-HD

Ⓐセット:CZ-662C+CZ-603D……定価¥492,800▶特価(現金価格はお電話下さい)

12回	32,800	24回	17,200	36回	11,800	48回	9,200	60回	7,600

Ⓑセット:CZ-662C+CZ-602D……定価¥507,800▶特価(現金価格はお電話下さい)

12回	34,200	24回	17.900	36回	12,300	48回	9,600	60回	8,000

Ⓒセット:CZ-662C+CZ-612D……定価¥527,800▶特価(現金価格はお電話下さい)

12回	35,700	24回	18,700	36回	12,800	48回	10,000	60回	8,300

Ⓓセット:CZ-662C+CU-21CD……定価¥547,800▶特価(現金価格はお電話下さい)

12回	36,100	24回	18,900	36回	13,000	48回	10,100	60回	8,400

X68000ACE-HD〜P&Aスペシャルセット=限定誌上販売!!

X-68000ACE-HDセット(台数限定)

●CZ-611C(本体)
●CZ-603D(モニター)
●CZ-8NJ2(CYBER STICK)

+ (●ディスケット10枚 ●ゲーム ●送料、消費税込み) ⇒ 定価¥508,400 P&A超特価 価格はお電話下さい

12回	28,700	24回	15,000	36回	10,300	48回	8,000	60回	6,700

モニターをCZ-602D(定価¥99,800)に変更の場合

12回	30,100	24回	15,700	36回	10,800	48回	8,400	60回	7,000

●CZ-612D(定価¥119,800)に変更の場合

12回	31,300	24回	16,400	36回	11,300	48回	8,700	60回	7,300

●CZ-611D(定価¥145,000)に変更の場合

12回	30,700	24回	16,100	36回	11,000	48回	8,600	60回	7,100

(ボーナス併用も有ります。TEL下さい。)

●本広告の掲載の商品の価格については、消費税は含まれておりません。
●営業時間=平日AM10:00〜PM8:00、日祭AM10:00〜PM8:00

回～60回払いまでOK.!!

★頭金なし！★即日発送

P&AがズバリI超特価セールでご奉仕!!

立寄り下さい。専門係員が説明いたします。
価で受付します。詳しくは電話にてお問合せ下さい。
価の20%引きOK！ TELください。

全国通販

超特価でクレジットが組める!!

X68000用ソフトコーナー（送料1ヶ～5ヶまで￥500）

品名	定価	特価
Z's STAFF PRO68K Ver2.0（ツァイト）	定価￥58,000	特価￥40,600
C-TRACE68（キャスト）	定価￥68,000	特価￥50,300
彩CRONE（アンス・コンサルタンツ）	定価￥58,000	特価￥44,600
アニメキット（アンス・コンサルタンツ）	定価￥5,000	特価￥4,000
テラッツォ（ハミングバード）	定価￥19,800	特価￥15,800
G-68K（OH! BISINESS）	定価￥14,800	特価￥11,400
KAMIKAZE（サムシング・グッド）	定価￥68,800	特価￥46,800
EW&EI（イースト）	定価￥38,800	特価￥28,800
C&Professional Pack（マイクロウェアジャパン）	定価￥58,800	特価￥46,000
Final Ver3.2（エーエスピー）	定価￥38,000	特価￥30,000
DATA PRO68K CZ220BS	定価￥58,000	P&A特価
CARD PRO68K CZ226BS	定価￥29,800	TEL下さい。!
C compiler PRO68K CZ211LS	定価￥39,800	特価￥32,000
OS-9/X68000 CZ219SS	定価￥29,800	P&A特価 TEL下さい。
AI-68K CZ234LS	定価￥188,000	特価￥143,000
THE福袋V2.0 CZ224LS	定価￥9,980	特価￥18,000
SOUND PRO68K	定価￥15,800	特価￥12,500
MUSIC PRO68K CZ213MS	定価￥15,800	P&A特価 TEL下さい。
Sampling PRO68K CZ215MS	定価￥17,800	特価￥14,000
MUSIC-studio PRO68K 237MS	定価￥15,800	P&A特価 TEL下さい。
MUSIC-PRO68K〔MIDI〕 247MS	定価￥18,800	特価￥22,000
New-print Shop 221HS	定価￥19,800	P&A特価
Communication 223CS	定価￥19,800	TEL下さい。!

ゲームソフト（1ヶ～20ヶまで送料￥500）

X68000用

品名	定価	特価
Ⓐ源平討魔伝（電波新聞社）	定価￥7,800	特価￥6,200
Ⓑドラゴンスピリット（電波新聞社）	定価￥8,800	特価￥7,000
Ⓒスペースハリアー（電波新聞社）	定価￥6,800	特価￥5,400
Ⓓ熱血高校ドッジボール部（SHARP）	定価￥7,800	P&A超特価
Ⓔ沙羅曼蛇（SHARP）	定価￥8,800	P&A超特価
Ⓕフルスロットル（SHARP）	定価￥8,800	P&A超特価
Ⓖ琥珀色の遺言（リバーヒルソフト）	定価￥9,800	特価￥7,800
Ⓗザ・スーパーラスベガス（日本デグスタ）	定価￥12,800	特価￥10,200
Ⓘマイト・アンド・マジック（スタークラフト）	定価￥9,800	特価￥7,800
Ⓙザ・リターン・オブ・イシター（SPS）	定価￥7,800	特価￥6,200
Ⓚ信長の野望（全国版）（KOEI）	定価￥9,800	特価￥7,800
Ⓛ麻雀悟空（シャノアール）	定価￥7,800	特価￥6,200
ⓂマーダークラブDX（リバーヒルソフト）	定価￥7,800	特価￥6,200
Ⓝザキングオブシカゴ（ボーステック）	定価￥12,800	特価￥10,200
Ⓞ今夜も朝までパワフルまあじゃん2（dB-SOFT）	定価￥7,800	特価￥6,200
Ⓟ三国志（光栄）	定価￥14,800	特価￥12,000

モデムコーナー （送料￥1,000）

品名	定価	特価
ⒶMD-2400B（オムロン）	定価￥49,800	特価￥36,000
ⒷMD-2400F（オムロン）	定価￥59,800	特価￥42,000
ⒸPV-A2400MNP4（アイワ）	定価￥46,800	特価￥35,000
ⒹPV-A24MNP5（アイワ）	定価￥54,800	特価￥41,000

周辺機器コーナー（送料￥1,000）

品名	定価	特価
ⒶCZ-8NSI	定価￥188,000	特価TEL下さい。
ⒷCZ-6VTI	定価￥69,800	特価￥54,000
ⒸCZ-6TU	定価￥33,100	特価TEL下さい。
ⒹBF-68PRO	定価￥19,800	特価￥15,500
ⒺCZ-6BEI	定価￥35,000	特価￥27,000
ⒻCZ-6BEIA	定価￥38,000	特価TEL下さい。
ⒼCZ-6BE2	定価￥79,800	特価TEL下さい。
ⒽCZ-6BE4	定価￥138,000	特価￥107,000
ⒾCZ-6BFI	定価￥49,800	特価TEL下さい。
ⒿCZ-6BPI	定価￥79,800	特価￥62,000
ⓀCZ-6BMI	定価￥26,800	特価TEL下さい。
ⓁCZ-6EBI	定価￥88,000	特価TEL下さい。
ⓂAN-S100	定価￥36,600	特価￥28,500
ⓃCZ-6SDI	定価￥44,800	特価￥35,000
ⓄCZ-8PC3	定価￥65,800	P&A超特価 TEL下さい。
ⓅCZ-8PC4	定価￥99,800	P&A超特価 TEL下さい。
ⓆCZ-8PK7	定価￥122,000	P&A超特価 TEL下さい。
ⓇCZ-8PK8	定価￥152,000	P&A超特価 TEL下さい。
ⓈCZ-8PK9	定価￥89,800	P&A超特価 TEL下さい。
ⓉCZ-6PVI	定価￥198,000	特価￥155,000
ⓊIO-735X	定価￥248,000	特価TEL下さい。
ⓋCZ-8BSI	定価￥23,800	特価￥19,000

P & A 特選パソコンラック （送料無料）移動自由（キャスター付）

中古パソコン　送料￥2,000

品名	価格	品名	価格	品名	価格
●X-68000セット	¥220,000	●CZ-856C	¥55,000	●CU-14AG2	¥40,000
●X-68000ACEセット	¥250,000	●CZ-870C	¥65,000	●CU-14H2	¥40,000
●X1ターボZセット	¥110,000	●CZ-881C	¥75,000	●CZ-8PC2	¥35,000
●X-1G/30セット	¥49,000	●CZ-820D	¥20,000	●CZ-8PK6	¥42,000
●CZ-822C	¥25,000	●CU-14GB	¥15,000		
●CZ-830C	¥35,000	●CU-14BD	¥35,000		

中古パソコンはＰ＆Ａにおまかせ.!!

その場で高価現金買取り・高価下取りOK.!!

■まずはお電話下さい。
03-651-1884
FAX：03-651-0141

■下取り・買取りでお急ぎの方、直接当社に来店、または、宅急便にてお送り下さい。

●下取りの場合……価格は常に変動していますので査定額をお電話で確認して下さい。（差額は、P&A超低金利クレジットをご利用下さい。）

●買取りの場合……現品が着き次第、2日以内に買取り金額を連絡し、振込み、又は書留でお送り致します。

●近郊の方は、P&A本店まで、直接お持ち下さい。
即金にて、￥1,000,000までお支払い致します。

通信販売お申し込みのご案内

〔現金一括でお申し込みの方〕
●商品名およびお客様の住所・氏名・電話番号をご記入の上、代金を当社まで、現金書留でお送りください。（プリンター・フロッピーの場合、本体使用機種名を明記のこと）

〔銀行振込でお申し込みの方〕
●銀行振込ご希望の方は必ずお振込みの前にお電話にてお客様のご住所・お名前・商品名等をお知らせください。
（電信扱いでお振込み下さい。）

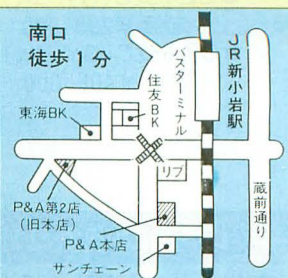
〔振込先〕住友銀行　新小岩支店
当No.263914 ㈱ピー・アンド・エー

〔クレジットでお申し込みの方〕
●電話にてお申し込みください。クレジット申し込み用紙をお送りいたしますので、ご記入の上、当社までお送りください。
●現金特別価格でクレジットが利用できます。残金のみに金利がかかります。
●1回～60回払いまで出来ます。但し、1回のお支払い額は3,000円以上。

アフターサービス万全
全商品保証付。専門の担当者がお客様の立場で対応します。
初期不良、輸送トラブル etc.
万が一初期不良、輸送トラブルが発生しました際には、即交換させていただきます。

●定休日/毎週水曜日＝第3水曜・木曜は連休とさせていただきます（祭日の場合は翌日になります）

超低金利クレジット率

回数	1	3	6	10	12	15	18	24	36	48	60
利率(%)	1.5	2.0	3.0	4.5	4.5	7.5	9.0	9.5	13	17	22

●マイコン
●ビデオ
●ビデオテープ

株式会社ピー・アンド・エー
〒124 東京都葛飾区新小岩2丁目1番地19号
☎03-651-0148（代）
FAX. 03-651-0141

営業時間
平日AM10：00～PM8：00
日祭AM10：00～PM8：00

●現金書留及び銀行振込でお申し込みの方は、上記商品の料金に3%加算の上でお申し込み下さい。詳しくは、お電話でお問い合せ下さい。

micro Computer入門

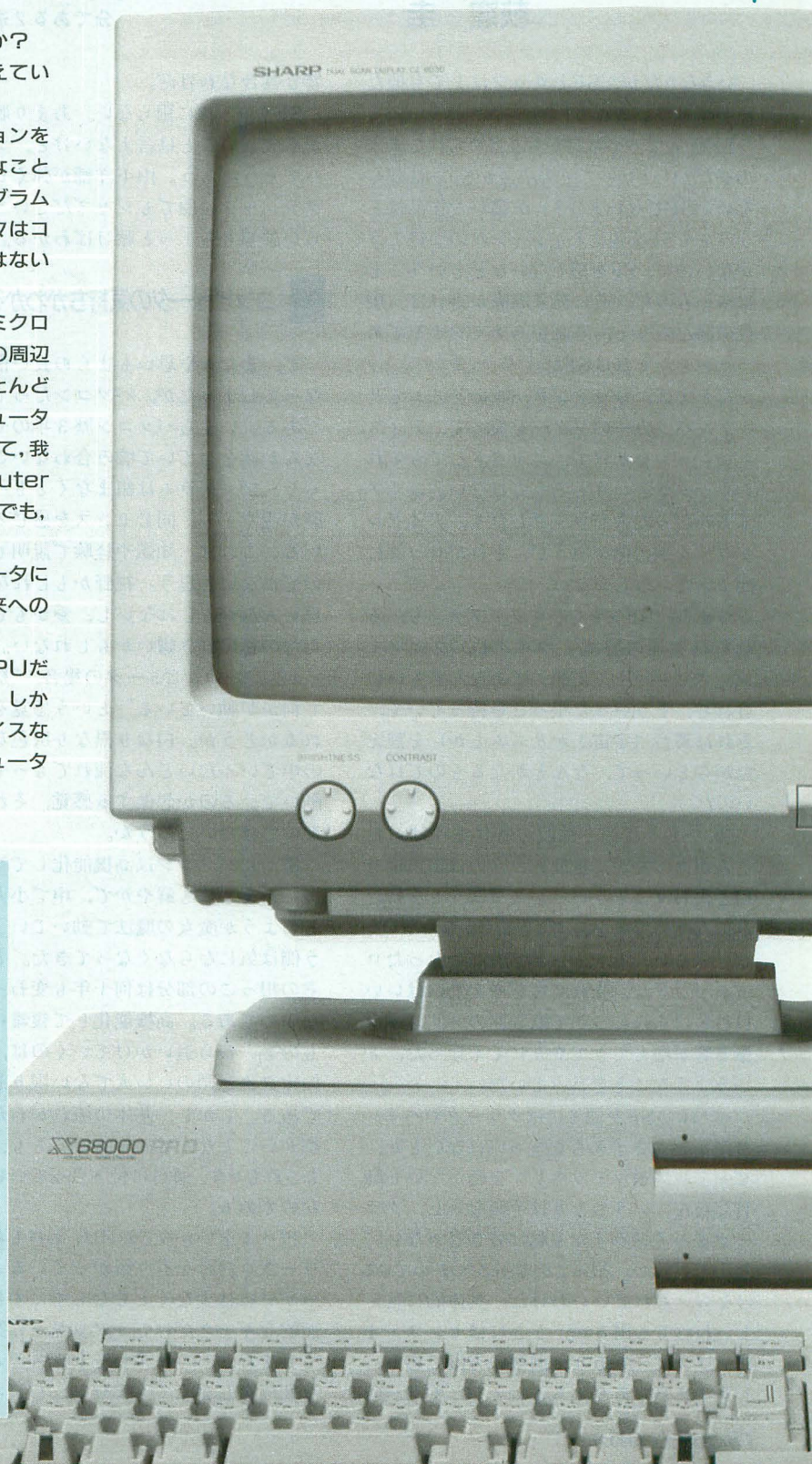
──「コンピュータってなに?」
こんな素朴な疑問にあなたならどう答えるだろうか?
ここで一度起点に帰って，我々がただ漠然ととらえているコンピュータというものを見つめ直してみたい。
コンピュータを使う，ということがアプリケーションを使う，ということと同義ならば，なにもこのようなことを考える必要はない。だが，コンピュータでプログラムをする人がマニアと呼ばれる風潮のなかでは，我々はコンピュータとはなにかということを見失いがちではないだろうか。
今回扱うのは“micro” Computerのもっともミクロな，もっとも核になる部分，すなわちCPUとその周辺だ。現在のパソコンを特徴づける多彩な機能のほとんどはコントローラLSIの力によるものだが，コンピュータ全体を規定するのは依然としてCPUである。そして，我我が使っているパソコンも中身はmicro Computerといわれるものにほかならない。規模は“micro”でも，その実体はまぎれもなく“Computer”なのだ。
Z80ではなく80286でも68000でもなく，コンピュータにおけるCPUを再認識することから，さらなる未来へのパースペクティブが明らかになるかもしれない。
もちろん，本来，micro Computer入門はCPUだけにとどまらない。これは第1歩にすぎないのだ。しかし，この1歩はOS，言語，ユーザーインタフェイスなど，新しい切り口から，より広い意味でのコンピュータ入門につながっていくに違いない。

CONTENTS

micro computer 入門

0と1の大行進

コンピュータの根っこ

古今東西，コンピュータの基本は0と1の2進数と相場が決まっています。それは，パソコンであろうと，大型機であろうと変わることがありません。コンピュータを理解するポイントは，根っこの部分である2進表現のスピリットを感じ取ることかもしれません。

Ogikubo Kei
荻窪 圭

いきなりだが，SFとロックはとても似た面を持っている。

SF仕立ての小道具がちりばめられた未来の話だからといってその小説がSFとは限らない。SFでない人(?)が書いた作品はどうみてもSF小説としかジャンルのつけようがないのに，SFを感じないなどということは多いものだ。逆に筒井康隆が書けば，現代の話だろうと，考証がムチャクチャであろうとも，それはSFだ。

たとえば，ロックンロールのリズムでリッケンバッカーのギターを使って，ディストーションをかけて，シャウトしてもそれがロックかというとそうではない。よくアイドルだった人がロックしたり，アイドルがロック調の曲を歌うが，それがロックに聞こえたためしがない。

つまり，SFファンもロックファンも，それぞれ共通の感覚，マインド，フェロモン，スピリット，幻想，まあなんでもいいけれど，そういった根っこを持っていて，それは舞台（宇宙とかリズムとか）を整えたからといって，なんとかなるものではないのだ。

昔，SFの浸透と拡散の時代と呼ばれたころがあったが，拡散というのは市民権を得る代わりにそういったスピリットを持たない人々の侵入を許すことにほかならなかった。あるヒットチャートで有名になったロックバンドが，売れて客が増えたのはいいけれど，それにつれて歌謡曲のノリで手を振る客が増えてきてやりにくくなった，というようなことを言っていた。

べつに，SFを語るにはクラークやハインラインからまず入らなきゃいけないとか，ビートルズやツェッペリンを聴かないと駄目だなんていうつもりは毛頭ないし，クラークもハインラインも私は好きじゃない。なんというか，根っこの部分がつかめているのなら，それでいいわけだ。具体的に言えといわれても困るが，ときにはセンス・オブ・ワンダーであり，洞察の方向であり，ドライブ感であり，ちょっとしたギターの響き具合なわけだ。

SFやロックに限らない。あまり聴かないから大きいことは言えないけど，ジャズだってそうだろう。山下洋輔が弾くピアノはクラシックの曲でもジャズだということくらい誰でもちょっと聴けばわかる。

コンピュータの気持ちがわかる?

で，またもや思いもよらぬ長い前置きになってしまったが，パソコンだって，そうである。いくらパソコン歴3年のやつでもなんか話をしていて噛み合わないこともあるし，プログラムは組まなくともパソコン歴が浅くとも，同じセンスを感じるやつもいる。これは，知識や経験で説明できるものではないと思う。視野かもしれないし，思い入れかもしれないし，愛かもしれない。ただの私の好き嫌いかもしれない。

が，ことコンピュータの場合，“機械の中で何かが動いている”という感覚を感じられるかどうか。白なり黒なり灰色なりの箱の中でいったいどんな流れでもって小人が動いているのか想像する感覚，それが大事なのではないだろうか。

確かにパソコンは高機能化して複雑になってあまりにも鮮やかで，中で小人が動いていようが魔女の魔法で動いていようが使う側は気にならなくなってきた。とはいえ，その根っこの部分は何十年も変わってはいないのである。高機能化して複雑になったものを一から追いかけていくのは，よほどのオタクか頭のいい人でない限り愚の骨頂である。しかし，基本の流れがわかれば，細かいことなんてわからなくとも，支配しもされもせず，適当に付き合っていけるものなのである。

根っこをつかんでおけば，わけもわからずワープロだけを有り難がっているようなユーザーには少なくともならずにすむし，表面的なギャグだけでオタッキーに笑わなくてもすむし，エキスパートシステムに無謀な期待をしなくてもすむし，パソコンも拡散しないで浸透していってくれるのではないか。夢のまた夢，また夢の夢。

指は10本あるけれど……

コンピュータの根っこが理解されにくいのは，98が悪いのでも，MS-DOSが悪いのでも，一太郎が悪いのでも，IBMが悪いのでも（よくもないけれど）なくて，その抽象性にある，と思うんだなあ。

*

時は紀元前3000年頃の，そうだな，日本としておこう。縄文人のある集落の話だ。まだ神武天皇さんもいない，新石器時代。

ここにひとりの天才がいた。彼は火葬場の管理をしていたので，斎場と呼ばれていた（いやな予感がするだろう）。縄文人が火葬なんかしていたかって？ そんな昔のことはわからない。まあ，細かいことは抜きにして，当時の人は10までしか数を数えられなかった。指が10本だったからだ。

当時，10以上の数を数えるときは，その辺にいるやつをひとり捕まえてきて両手を広げて立たせておき，それを10の単位とした。たとえば，23を表すには，両手を広げた男2人と，指を3本だけ出した男がひとり必要だったわけだ。彼らは山の中に住む狩猟民族だったのだが，1日に兎を何十羽*1も捕れたときにはその数を長老に伝えるのが大変だったわけだ。

ある日，斎場が長老のところへ寝不足の顔をして現れた。

「長老，数の数え方についてご相談があります」

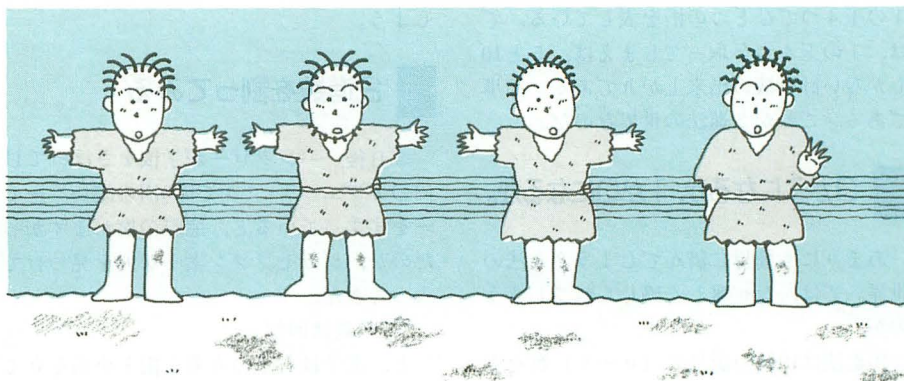
斎場は人差し指と中指を立てて見せ，

「兎がこれだけ捕れたとカクさんが言ったとしましょう。次にスケさんが来て，親指と小指を立て，兎がこれだけ捕れた，と言ったとしましょう」

斎場は親指と小指を立てて見せた。

「さて，長老。スケさんとカクさんではどちらがたくさん兎を捕ったのですか」

長老は斎場が何を言いたいのかわからず，

怪訝な顔をして「もちろん，同じだな」と，言った。斎場は我が意を得たり。
「長老，どちらも指の立ち方は違うのに同じ数とは，無駄ではないでしょうか」
「無駄？　それのどこがいけないんだ？」
「いけないのではありません。ただ，もし指の立て方で表す数が違うとなれば，兎を10羽（ここで斎場は両手を広げて見せ）よりたくさん捕ったときでも，ひとりで表せるのではないでしょうか」
長老は「うーむ」とうなったまま微動だにしない。やがて，
「いい方法はあるのか，斎場よ」
と，囁いた。
「もちろんでございます。指が立っていないときも使うのであります。まず，片手でご説明しましょう」
斎場は右手を握って，手の甲を長老に向けて突き出した。「これは1羽も捕れなかった情けないときの印でございます」
（著者注：おお，こんな昔にゼロの概念を見つけたものがあるとは！）
続いて，斎場は親指を立てた。
「1羽だけ捕れた不満なときの印はこれでございます」
「ほかの指ではいかんのか？」
「はい，順番というものがございますゆえ」
長老は真似をして右手親指を立ててみた。続いて，親指を引っ込め，人指し指を立てた。
「これが家族で食せる2羽の印だな」
「はい，さすが，長老，そのとおりです」
「わかった。これが3羽だろう」
長老は唇の端を僅かに歪め，中指だけを立てて見せた。斎場は予め答えを用意していたかのように，
「いえ，それでは，今までどおり10羽までしか数えられません。それは，4羽の印です」
長老は目をむいた。10秒ほど森の神へ祈り，続いて狩りの神へ祈った。
「3羽はこうです」
斎場は親指と人指し指を立てて見せた。
「難解だのう」
「いえ，とても簡単な仕組みです。ご説明しましょう」
斎場は予め抱えてきた甘い拳ほどの大きさの木の実を5つ2人の間に置いた。
「これを数えてみましょう」
斎場は左手で実をひとつ取り，右手の親指を立てた。
「これでひとつです」
続いて，もうひとつ左手に増やした。

「2つ目を取りましたので，右手の数を増やしましょう。私の考えた方法では，数は常に親指に対して加えられます。しかし，親指はすでに使われているので，加えられません。そこで，親指2つ分という意味で，人指し指を立てるのです。すると，親指は空きます」
斎場は右手人差し指を立てて親指を折った。左手で3つ目の木の実を取った。
「次は，親指が空いていますので，立てます。この，人差し指と親指の2本が立っているときが3個なのです。指は2本でも3個なのです」
さらに4つ目の木の実を左手に収めた。
「さらに，ひとつ増えましたので，親指に加えますが，またもや親指は使われております。そこで，親指を休ませまして，人差し指，ところが，人差し指も立っているではないですか。しかたがないので人差し指も休ませまして中指を立てます。これが4つを表す指です」
これを3日3晩繰り返し，斎場は辛抱強く長老に教え込んだ。
「私の考えた方法で数を数えれば，片手だけで3人と指1本分。両手をつなげますと，なんと，10人の10倍以上の数の兎でも数えられるのです」
「娘たちが山ほど抱えて帰ってくるくぬぎの実も数えられるというのか」
やがて，斎場の村では10年かけてこの記数法が普及し，その副作用でみんな指先が器用になった。指先の器用な人々はここが発祥だったわけだ。なお，隣の集落へも伝わったが，その集落の長老は不器用なために“薬指と親指だけ立てる”形がうまく作れなかった[*2]。よって，普及しなかった。
その後，この記数法は，大和朝廷が日本統一の旗のもと，この集落の末裔を平定するまで使われる。もうそのころには，元の意味を知るものもなく，ただどの数はどの指の形というパターンだけしか伝えられなかったため，この知恵はあっさりと失われた[*3]。

パンクローのバイナリ講座

「と，いうわけだね。僕はこの斎場の末裔なのさ。へっへ」
と，斎場パンクローが得意気に鼻を鳴らした。
非常に私としては心苦しいのであるが，いやな予感のとおり，またもや懲りずに斎場パンクローとジョセフソン素子の登場である。
時は，前回のお話を遡ること5年，2人とも高校へ入ったばかり，好奇心旺盛な斎場パンクローが，まだコンピュータといえばゲームマシンか銀行の中で金勘定をする拝金主義の機械としか認識していないジョセフソン素子に計算機の原理を悪戦苦闘して説明するところから始まるわけだ。
「今のつまんない話が何だって言うの？」
「2進法だよ。2進法は僕らが使っている10進法に比べて，指だけで1000以上も数えられる位単純で簡単な方法だっていう話」
パンクローは，あと1年もすれば素子に追い抜かれて馬鹿にされるとも知らず，得意気なのであった。
素子はくやしげに，指を立てたり折ったりしている。
「いいか。右手の親指が下1桁で，人差し指が2桁目，小指が5桁目だと考えて，考えた？　5と10しか数字がない世の中でも考えてみればいいんだ」
5と10しか数字のない世の中。諸君はソロバンという手動式計算支援機械を知っているだろうか。あれは，5の玉ひとつと，

＊1　みんな知っていると思うが，兎はいちわ，にわと数えるのだぞ。

＊2　実は私もだ。指がツリそうになる。こんなこと始めるんじゃなかった。

＊3　だいたいにおいて，知恵というのはこうして失われるものだ。精神が滅び，形が残ったもののなんと多いことか。

1の玉4つでひとつの桁を表している。では，1の玉4つを取ってしまえば，5と10しかない計算機の出来上がりである。簡単である。これが2進法の世界なのだ。

0が1になるか，1が0になるか。

あまりにも簡単に済んでしまう2進法の世界。では，なぜ難しく感じられてしまうのか。

10進法は10個の記号で（0～9）数を表すが，2進法は2個しか使わない。だから，覚えることは少ない。しかし，記号が少ないということは，同じ数を表すのにも沢山の桁が必要になるということであり，把握しづらくなっていく。

人は常に複雑な概念を必要とするものが現れると，それを表す言葉に置き換えて対処してきた。CDなんてコンパクトディスクの略だが，コンパクトディスクと聞いても，それだけでは何のことやらわからない。光学読み取り式円盤といっても，その言葉には中に記録されているだろう音楽に思いを寄せることは不可能だ。つまり，CDという単語には普段何気なく使っている以上のものが込められているということで，人は常にこのようにしてひとつの言葉なりものなりに多数のものを込めることによって，複雑化・多様化する流れに対処してきたのである。

コンピュータはその人の営みに逆行するかのように，0と1の支配するもっとも単純なものしか受け付けてくれない。だから，コンピュータは非人間的なるものの代表であり，コンピュータと対峙することは人間が楽をしようとあみだした“言語・概念生成能力”との対決にほかならないのである。

だから，いつまでたってもコンピュータが世間の人に正しい評価をもらえないのである。この対決を具体的に行っているのが人工知能の世界だ。人工知能の研究なんて，複雑化・多様化した概念を閉じ込めた言葉を元に復元しようとする非人間的な試みなのである。

しかし，本誌読者諸君なら肌で感じとっているだろうことと思われるが，こういった非人間的とも思える“分解”の作業は，泥臭くて生臭い人間的なさまざまなものに比べ，極めて純粋な作業である。だからこそ面白いのであり，複雑に組み合わされ，人々の目から覆い隠された事物を分解して中身を探りたくなる気持ちもわかるというものだ。

と，いうわけで，もう少し2進法の話をしよう。

2進数を割ってみる

3日後。パンクローが学校をさぼって目白のカフェ・ラ・ミルで500円の濃いコーヒーをすすっていると，窓際の席がまずかったのか，ジョセフソン素子が彼を見つけて入ってきた。

「この数は何だ」

と，素子は右手の人差し指と小指を立てて見せた。

「もしかして，3日も指の体操ばかりしてたんじゃないだろうな」

「してたって，いいじゃない。あ，私，紅茶とシフォンケーキね」

ウェイターを軽くあしらうと，素子はピュアな2進法が気に入ったらしく，

「じゃあこれは？」

と，左手と右手を出して指を何本か立てて見せた。パンクローは2進法の理屈は知っていても滅多に使わないし，そんなパズルに価値を見出さなかったが，素子はそうではなかった。

「やーい。308だよ」

パンクローは面白くない。

「じゃあそれを4で割ってみろ」

さあ，素子は悩んだ。パンクローが簡単に言うからには，すぐに答えが出るような秘密があるに違いない。パンクローは勝ち誇って禁煙パイポを吸っている。暗算で計算し，その結果をまた2進に直すなんて，素子のプライドは許さない。

「くだらねえことで悩んでないで，さっさと食え。そのケーキ，食っちまうぞ」

「ちょっと待って。待ってったらあ」

そう言いつつも，素子の目は自分の2進計算機。つまり指から離れていない。

「おまえなあ。500割る100はいくつだ？」

「5」

「それとどこが違うんだ？」

「あ，そっかそっか。4は100（2進法だから，イチゼロゼロね）だから，2桁減らせばいいわけね」

「そのとおり。ケーキは教授料ね」

「いいわよ。だから，ちゃんと教えて？足し算から」

パンクローは拍子抜けして，ひと言教えた。

2進法演算のいろいろ

「はい，右手を出して。左手も出して。右手さんに左手さんの数を足してあげましょう」

と，パンクローは皮肉たっぷりに言い，素子はそうして，指がつりそうになった。

「じゃあ，普通の筆算みたいに，桁を揃えて，はい，下の桁から足していきましょう，と。数字が変わらないか，繰り上がるかしかないから簡単だろう。俺なんか，8＋7が覚えられなくて苦労したのに，2進だったらそんな苦労いらないもんな。2進数ってすごい」

パンクローは勝手にうなずいている。

「じゃあ，掛け算は？　九九みたいなのはないの？」

「あるよ。はい，インイチがイチ。おしまい」

「あ，そう。そういうもんなの」

「そうそう。九九なんて10進法が複雑すぎて計算が面倒臭い。面倒ならいっそ覚えてしまえ！　と，考え出されたものなのだ，絶対」

素子は呆れて可愛いメモ帳を取りだした。パンクローが乱暴な字で

```
      1100111
  ×     11101
  ─────────────
```

と，書いた。

「どうやるの？」

「10進法でも2進法でもインイチはイチなんだから，普通にすればいいに決まってるじゃないか。最後に足すときだけ，2進法でやりゃあいいんだ」

「あ，そう。これでいいの？」

```
      1100111
  ×     11101
  ─────────────
      1100111
    1100111
   1100111
  1100111
```

「で，これを足せばいいのね」

素子はこれをじっと見つめて悩み，

「ねえ，これ足すのって，面倒臭い」

「当たり前だ。だから，人様がやらないで，コンピュータにやらせるんじゃねえか」

「あ，そうか」

さっき，人間の営みひとつに，複雑化・多様化してしまった概念をひとつの新しい言葉で置き換えることがあるという話をした気がする。その営みの弊害を挙げよう。

元の概念を知らずに新しい言葉だけを見たとき，それが何を示し，何を語っているのか“さっぱり，パッパラパー”なのだ。ちゃんと言葉の向こうの精神を知ろうと思ったら，せっかく作られた新しい言葉から

元の“単純な言葉を複雑に組み合わせて語られた概念”へと逆行していかねばならない[*4]。ひどい場合には，結果にたどりつくために本を3冊読まねばならなかったり，さらにその本の言葉を理解するために本を読まねばならなかったりで，終わってみたら“たくさん本を読んだ”という事実だけが残って，頭が飽和して煮詰まって“結局よくわからなかった”となることが非常に多い。とっても多い。そんなわけのわかんないことばっか書いてある本なんて作るなよな（と，学生の頃はこんなことばっか思っていた）。

思うに，頭のいい人というのは，“何が書いてあるかわからない，まるで読まれることを拒否しているかのような書物”から，何となくポイントをつかんでしまえる能力を持った人のことではあるまいか。重要なのは知識や記憶力ではないのである。そういったことは，コンピュータにまかせ……られるようになったらいいな，と。

さて，話はもどって，2進法の話は少し具体性を帯びてくる。

1と0だけでどこまでできるというのか

「2進法はわかったけど，小数とか負の数とかはどうするの？」

「そりゃあ，2倍するごとに桁が倍になっていくんだから，半分になるごとに桁が減るんだろ。この考えでいくと，2進数の0.1は10進の0.5で，0.01はえーと，0.5の半分なのさ」

「0.25ね。じゃあその半分は0.125ね」

暗算の嫌いなパンクローはどこでこの話題を切り替えようかと思い始めていた。

「もしかして，2進数の0.111は，0.5+0.25+0.125を計算しないと10進数にならないの？」

「そのとおり。111を半分にして11.1でさらに半分にしてもう1回半分にすれば0.111だから，7を3回半分にすればいいわけだ」

「えーと，0.875かしら」

「そんなもん知らん，勝手に計算しろ」

「もしかして，10進の0.9を2進数にしようと思ったら，0.1111111……っていくつ1が続くの？」

「んなもん知るか。俺が知ってるのは小数になると，完璧な10進→2進変換はできないっていうことだけ」

「えー，じゃあ，コンピュータって嘘ついてるの？　だって，どのコンピュータも中では2進数なんでしょ。だったら，コンピュータが計算した割り算の結果は間違ってるんじゃん」

「おまえさんにはわからない人間様の深い知恵がそうはなんないように工夫してるのさ」

「ずるーい」

ここで素子はパンクローを困らせるネタを閃いた。

「じゃあ，0と1しかない世界なのに，どうして小数点があるの？　小数点は0でも1でもないじゃない」

「うっ」

「ついでに，−1とか−01101はどうしてるの。マイナスは0でも1でもないわよ」

「むぐぐ」

パンクローは困った。言われてみれば確かにそうだ。

「それは，おまえ，おまえにはわからない人間の深い知恵がだなあ……」

「あ，そう。自分で調べるからいいわ」

こうしてパンクローは素子の尊敬を勝ち取るチャンスを逃したのであった。

こんな調子なので，荻窪圭が仕方なく解説するのである。

答えは簡単。“コンピュータはバカである”ということだ。バカだから，小数なんて概念も負の数なんて概念もない。

しかし，考えてみるといい。我々は筆算をするとき，小数点のことをどう扱うだろうか。まず，紙に式を書くとき，桁を合わせるために使う。そして，計算の最中はまったく無視して，終わってから，小数点を打つだろう。それで困りはしない。

と，いうわけで，コンピュータがバカだから小数点を無視して計算したとしても困らないわけである。小数点の位置さえどこかで把握していればいいわけなのだ。

コンピュータで小数を扱う一番簡単な方法は，何桁目が小数点か予め決めておく方式[*5]である。たとえば，2進10桁の数があるとして（指10本分だね），下から5桁目と4桁目の間（右手を下の桁側とすると，右手小指と薬指の間だ）にあるよ，ということにしておく。誰がしておくかというと，私だ。

それでもって，2進自然数（?）の1101を使うときは，小数点位置を合わせるために後ろに4つのゼロをくっつけてやればいい。こんな調子で計算をする。コンピュータはバカだから小数点があるなんて思いもしないで普通に計算をする。結果が出る。

私はその結果を見て，最初に小数点と決めた場所に小数点をくっつけて答えを見ればいい。答えが11011100だったとしたら，それは1101.11だと思えばいいわけだ。

原理なんて，こんなもんだ。

ついでに，負の数の場合もおんなじである。コンピュータはバカなのだ。

たとえば，コンピュータが4桁までしか計算ができないとしよう。最初は0が入っている。4桁だから，“0000”だね。ここから1を引こう。人間から見ると，0から1を引くわけだから，−1なのはわかりきっている。コンピュータはバカだから，0000から1を引く。

まず下の桁から1を引くが，0だからできない。そこで上の桁から1を借りてきて下1桁は1だ。しかし，2番目の桁も0だから1を貸すことができない。そこで，さらに上の桁から1を借りてくる。しかし，3番目の桁も0だから1を貸せない。そこで，4番目の桁から1を借りる。4番目の桁は更に上から1を借りるが，4桁しか計算できないので，ここで“お手上げ”となり，計算は終わる。

すると，結果として借りっぱなしで返せない数が“1111”と4つも残ったわけだ[*6]。簡単だね。ここで誰かが考えたわけだ。

＊4　プログラムを解説したり，コンピュータをすみずみまでしゃぶろうという試みもこれに他ならない。だから人のプログラムを読むのは嫌いなのだ。特に16進のダンプを追うのは（逆アセンブルしてあっても）多大なるパワーを必要とする。パワーのある人ってすごいな。

＊5　これを固定小数点という。小数点の位置が固定だからで，ソロバンの足し算・引き算もそうだ。ソロバンほど桁が多ければよいが，コンピュータでやるとそんな大きな桁数はとれないので，左端とか右端に無駄ができたり，はみ出したりする。それでも固定小数点方式の10進演算（2進だと誤差が出るので，無理やり10進のまま計算するのだ）はCOBOLやPL/Iなどでよく使われる。反対に，浮動小数点という方式（パソコンの小数計算はみんなこれだ）もある。

「さあ困った。−1も15（2進の1111は10進で15だ）も同じ結果になってしまった。よし，この際細かいことに目をつぶって，一番上の桁が1だったら負の数ということにしよう！」

こんなものである。例によってコンピュータはバカだから素直に計算をし，その結果を人間が勝手に負の数だとしてしまったりするのだ。それだと足しすぎて一番上が1になったのか引きすぎてなったのかわからなくなったりするのだが[*7]，それはコンピュータにとってはどうでもいいことなのである。人間が気にすればいいだけだ。

このようにして，小数だろうが，負の数だろうが，コンピュータは全然気にしないでいつもの調子で足したり引いたりいろいろと1と0の世界で戯れているのである。ああ，優雅なことで。

コンピュータは0と1の大行進

今度は権威を失墜したパンクロー復活の話である。

「ねえ，パンクロー。そもそも0と1がさあ，どうしてどうなったらプログラムとかいう代物になるわけ？」

パンクロー君も今回はしっかり勉強をしてきているので大丈夫とばかり，眠い目をこすって。

「2進数の8桁がひとつの単位だと考えてみるんだ。コンピュータさんを作った人は，8桁単位でなんでも処理するようにするからね[*8]。どーして？　なんて聞かないように。とりあえず，そうなのだ」

「ASCIIコードも1文字8ビットだもんね」

素子だって勉強しているのである。

「うーん。話が早い。それで，その8ビットがずらーっと並んでいる，まあ，横8桁の幅を持った紙テープがあると思ってちょ。それが実に長いテープで，何メートルもあるんだ。まあ，1個の枠を1辺2ミリとすると，幅が1.6センチで長さが65536行として，131.072メートルね。なんで65536かというと，8ビットが2つで16ビットで，16桁の2進数は10進だと65536通り（0〜65535）だからね」

「よくそんな数字を覚えられたわね」

X68000ユーザーには呪文のように馴染み深い数字である。

「で，コンピュータさんの脳味噌君[*9]とその紙テープがつながっているとしよう。脳味噌君はその紙テープに書いてある8桁の1と0を同時に読んだり書いたりする機能があるんだ。ついでに，視野が狭くて一度に1行しか見られないんだけど，今何行目を見ているかがわかって，ついでに1行ずつテープをずらすことができる。さらには，脳味噌君は俺たちが文章を左から右に読むように，勝手に1行ずつ進んでいくんだ」

「その紙テープにいろいろと書いてあるわけね」

「そうそう。そこにいろいろと命令が書いてあったり数字が書いてあったりするんだ。実際はそれを紙テープじゃなくて電気でやってるんだけどね」

「そのテープのお尻と頭がつながっていて，さらにそれがメビウスの輪みたいに1回ひねってあると面白いね。4次元の計算でもしてくれないかしら」

「………」

だいたい原理はこんなものである（さっきも同じようなことを言ったなあ。ま，いいか）。で，その紙テープの何が命令で何がデータかという問題がある。コンピュータにとってみればどっちだって0と1の並びにすぎないわけだから，予めいろいろと約束ごとを決めておかねばならない。それはもう作った人の自由である。8ビットであったり，16ビットであったり，32ビットであったりするわけだが，とりあえず今の普通のコンピュータさんは1行に8桁しか書かれないという原則[*10]と，どんなコンピュータでも電源を入れたら1行目（この世界では数字は0から数えるので，0行目といおう）から読み始めるという鉄則[*11]は守っているはずだ。

この紙テープをメモリ，8ビットを1バイト，何行目かを表す言葉をアドレス，12行目をアドレスの12番地などなどと置き換えればあっという間にコンピュータの話となる。そんなものである。

コンピュータはどんなものでも0と1の並びの大行進で動いているのだ。

計算機とはもう呼べない

もともとコンピュータは戦争のとき，弾道計算を（大砲を撃ったとき，どの角度でどーんと撃てば目的の戦車を壊せるか）するために作られたものである（らしい）。だいたい，膨大な金がかかる新しい技術に国がポーンと金を払うのはそんなときくらいだ。

だから，コンピュータは電子計算機と訳され，計算が速い機械と認識されたのだが，英語の訳としては中国語のほうが一枚上手だったようだ。電脳としてしまったのだから。日本でも電子頭脳という言葉があったが，あれはあれで違うものを指していたから。

コンピュータは時代と共に計算をする機械ではなく，記号を扱う機械と化した。もともと，0と1の並びにすぎないのだから，それを数値とみなすか文字を表す記号とみなすかは人間の側の問題であるからして，コンピュータの知ったことではない。

今，コンピュータは記号を扱う機械から情報を扱う機械へと認識が変わってきている。中身（動作）は大して変わっていないのに，である。コンピュータの本来の姿は計算機ではなく，記号を扱う機械だと思う。記号の集まりを情報とみなすか否かは人間の役割だ。

今のコンピュータを作ったともいえる3人（フォン・ノイマン，アラン・チューリング，C.E.シャノン）の遺産を越えない限り“コンピュータはバカである”という事実は変わらない。昔も今もバカにものを教える（プログラミングのことだよ）のは大変だし，教える人間のほうもたいして頭がよくないときている。だからこそコンピュータを必要とするわけで，世の中うまくできているのか無駄に輪廻しているのかわからない。

今回はあまりにも入り口の話を試み，用語もほとんど出てない（はず）ので，物足りない人も多いだろうが，コンピュータの根っこは何時も0と1だということをもう一度心にとどめておいてほしい。

＊6　これを−1の2の補数表現という。世の解説書ではいきなり“2の補数とは”などと始まって，2の補数の求め方に突入してしまうので初心者にはわかりにくい。ちなみに，1の補数や10の補数もある。たとえば，車のトリップメーターなんか，0000でバックすると9999になってそこから減っていく。9999が10の補数表現で−1なのだ。

＊7　実際のところ，オーバーフロー（桁がはみ出す）したときや，正負がひっくり返ったときにはフラグというものが1になったり，0になったりするのだ。

＊8　でも，初期のコンピュータはそんなことはお構いなしに好き勝手やっていたけどね。ひとつの単位が70ビットとか（これはあとの記事のお楽しみ）。

＊9　言うまでもないことだけど，CPUね。

＊10　16ビットのCPUだって，一度に2つ分扱われている（偶数番地と奇数番地）だけで，ひとつの番地には1バイトである。

＊11　鉄則などと書いたが，実は8086と68000ではこの原則が当てはまらなかったりする。つまり，世の中の大半のコンピュータは普通じゃないことをしているのだ。困ったものだ。

micro computer 入門

マイクロコンピュータへの招待

初歩からのCPU物語

Misawa Kazuhiko
三沢 和彦

ノイマン型コンピュータの基本アーキテクチャを解説します。コンピュータアーキテクチャとはCPU回りのハードウェアだけではなくソフトウェアのインタフェイス部分も含みます。もっとも基本的な部分なので、最初から飛ばしすぎずしっかりと理解してください。

1 コンピュータの構成

1946年に開発された最初の自動計算機ENIACは真空管式でした。プログラムを組む代わりにたくさんのケーブルをつなぎ換えて制御を行うというもので，複雑なことをさせるのは難しく，その割にひと部屋分という大きさでした。

その後，半導体素子トランジスタの画期的な発明があり，さらにそれらのトランジスタを小さなパッケージに納めた集積回路(Integrated Circuit：IC)，その集積度を高めた大規模集積回路(Large Scale Integrated circuit：LSI)が次々に開発されるとともに，コンピュータはよりコンパクトに，より高機能になってきました。

そしてコンピュータの機能を1チップにまとめたマイクロプロセッサが開発されると，初めてマイクロコンピュータというものが現れるようになったのです。そういう意味では，今日私たちが使っているパーソナルコンピュータはかなり進化した形態のコンピュータだといえます。

しかし，どんなにハードウェアが進歩しても，計算機の基本的なアーキテクチャは1949年にフォン・ノイマンによって提案されたプログラム内蔵方式がいまなお主流となっています。これは計算の実行内容を記述したプログラムを記憶装置（メモリ）に順番に記憶させておき，それを中央処理装置(Central Processing Unit：CPU)が逐次読み出してきて実行していくというものです（ノイマン型コンピュータという)。

最初のアドレスにある命令を実行したら次のアドレスの命令を読み，以下，順に実行する……。この実に当たり前のような方式がノイマン型の本質です。ENIACのようにプログラムに応じてハードウェアを組み直すような方式に比べて柔軟性が高く，これによって初めてコンピュータにコンピュータらしい動作が期待できるようになったといってもいいでしょう。

ここでは，このノイマン型のアーキテクチャとそのハードウェアの構成を概説することにしていきます。

コンピュータの3大要素

ノイマン型の（すなわち，一般的に普及している）コンピュータは，大きく分けて次の3つの部分からなっています（図1参照)。

1) 記憶装置（メモリ）
2) 中央処理装置（CPU）
3) 入出力インタフェイス（I/O）

●メモリ

メモリはプログラムとデータを記憶させておくところです。メモリにはひとつずつ順番に番地（アドレス）が割り当ててあって，それぞれのアドレスに数値データが格納されています。

●CPU

CPUはコンピュータの中枢部で，命令やデータをメモリから読み込み，それを解読して実行したあとに，その結果を再びメモリに戻します。メモリに格納されている情

2進法の話

さて，電気や磁気を始め，自然界にあるものは2相（0/1など2つの状態）で安定するものがほとんどです（0/1/2の3つの状態で安定するものなど固体／液体／気体ぐらいでしょう)。人間が作るものもこれら物理現象などを利用していますので，どうしても0/1の状態を基本に設計されています。コンピュータでは電圧が一定値より高いかどうかで0/1を決めています。

これを制御するのはスイッチング素子としてのトランジスタです。トランジスタはアナログ回路では電圧の増幅に用いられますね。しかし，実際にはトランジスタで増幅できる電圧の帯域はそう広いわけではなく，増幅できないくらい高い電圧や増幅できないくらい低い電圧をかけてやると出力電圧ははっきりした安定状態になります。これを電圧のHigh, Lowの2値として位置づけているのです。通常はHighが1＝真，Lowが0＝偽として扱われます。

そしてトランジスタやコンデンサなどを組み合わせてAND, OR, NOTなどの論理素子(いわゆるゲート）が構成されます。「AかつBなら」とか「AまたはBなら」とかいう条件を扱うことができるようになると，これらの組み合わせで非常に複雑な論理も扱えるようになります。さらにこれらの論理素子を組み合わせて論理回路が作られ，ずっと進んで論理回路の親玉にあたるのがCPUなのです。CPUが基本的に0/1の信号しか扱えないのもこういった関係によるものです。

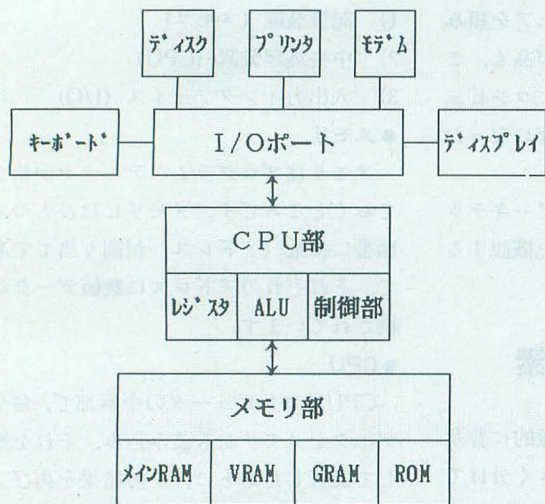
もしも，たとえば256段階で安定するトランジスタができたとすると，現在のメモリ1ビット分のメモリセルで1バイトの情報量が記憶できることになります。光素子などのなかには多状態で安定する素子も発見されていますので，いずれはこれらを使った2進法（2digitといいます）以外のHigh digitコンピュータができるかもしれません。

コンピュータ

広辞苑を引くと「コンピューター[computer] 計算機。主として電子計算機をいう」と載っている。要するに（誰でも知っているけど）マイクロコンピュータというのは超小型電子計算機であって，当然のようにあのちっこいシリコンのチップの上をばしばしと電子をあっち流しこっち流しして計算をしているわけですね。ということは昔コンピュータを作った人が電子の代わりにシリコンチップに水道管とバルブを作って（超精密機械だな）水で計算するようにしていればいまごろ「パソコン＝水力計算機」になっていたわけですね（実際，リレーを使ってガシャガシャと計算していた機械のことを電気計算機とも呼んでいたらしい)。うーん，それも面白かったかもしれない。

ずーっと昔，歯車をいくつも組み合わせ力一杯がっちゃんがっちゃん回して計算する手動機械式の計算機があったそうだ。うーん，人間が計算のエネルギーか，がんばるなあ。でも，それを元気計算機とはいわなかったと思う（ああ，くだらんネタを……)。

報について，それが実行すべきプログラムなのか，単なる数値データなのかは，それ自体には区別がなく，その場でCPUが判断します。CPUは，基本的な論理，算術演算のできる演算装置（Arithmetic and Logical Unit：ALU）も持っています。最新のコンピュータがこなすどんな複雑な処理であっても，この基本演算をいくつも組み合わせることによって行われているのです。

図1　コンピュータの基本構成

●I/O

コンピュータは人間によって使われる以上，人間とCPUとがやりとりする出入り口がなければなりません。これらの人間とのやりとりをする部分をI/Oといいます。入力装置としては，身近に使われているキーボードのほか，マウス，ジョイスティックなどがあります。出力装置には，ディスプレイ，プリンタが身近ですね。また，フロッピーディスクやハードディスクといった外部記憶装置も，メモリというよりはI/Oといってもよいかもしれません。

I/Oについても，複数の装置に対して各々アドレスを設定して，ある装置とデータをやりとりするときには，そのアドレスを指定するようになっています。Z80など，たいていのCPUではI/O制御専用の命令が用意されていることが多いのですが，どちらもアドレス指定してアクセスすることから，I/Oアドレスとメモリのアドレスとを分けないでメモリにアクセスするのと同じ方法でI/Oにアクセスする「メモリマップド I/O」という方式もあります。MC68000ではもっぱらこの方法がとられます。また，Z80を使った機種でもMZ-80K/Cなどはこの方式をとっています。

2 CPUのソフトウエア的アーキテクチャ

ノイマン型コンピュータはいま述べたような3つのブロックから成り立っていますが，すべての処理に対して命令を下すCPUのアーキテクチャがコンピュータ全体の構成を決めるといってもよいでしょう。そういう観点から，ここではCPUのアーキテクチャについて，詳しく考えてみることにします。

CPUの中身がどういう回路からなっているのか，そしてそれぞれの回路がどういう動作に対応しているのかを説明しながら，コンピュータの基礎に迫ってみたいと思います。

メモリのいろいろ

使用用途に従ってROM（Read Only Memory）とRAM（Random Access Memory）の2種類があります。

ROMは読み出し専用のメモリで，X1turboやMZ-2500，X68000などでは入出力関係の基本的な処理をサブルーチンのかたちで記憶させてあります。これは，基本的仕様を変えない限り，書き換える必要のないプログラムなので，このようなROMに入っているのです。

これに対してRAMは読み書き自由なメモリで，ユーザーが必要に応じて変更するプログラムやデータをロードして使います。また，処理の途中で一時的に退避させておきたいデータなども蓄えておきます。

また，システムによっては仮想記憶というものを使って実際に設置されているメモリ容量にかかわらず，巨大なアドレス空間（ハードディスクなどの補助記憶装置を使う）を見かけ上自由に使用することができる場合もあります。これは通常，CPUに付属するMMU（メモリマネージメントユニット）が管理する機能ですが，最近の32ビットCPUなどではCPU自体の機能として仮想記憶を扱うようです。

CPU

マイコンのCPUの主な系列にMC6800から始まって6802，6809，68000，68020，68030といったモトローラ開発の68系と8080，8085，Z80，8086，80286，80386のインテル系の80系（Z80はザイログのだけど8080上位コンパチだからね）がある。あとモステックの6502とかもあるけどだいたい昔からこの2つが主流だったんですね。昔のマイコン関係の本によく載っていた話なんだけど，68系と80系の決定的な違いはその生い立ちなんです。

むかしむかしの話。CPUとしては80系のいちばんの御先祖に当たる4ビットCPU4004が開発されました。が，実はこれは電卓用のLSIでした。当時，電卓の値下げ競争が続いて「こりゃかなわん。どんな電卓にでもプログラムして対応できる汎用チップを作って単価を下げよう」ということでできた石だったのです。

そしてそのあとインテルから8080が出され，直後にモトローラから6800が出されたわけですが，モトローラはこの6800を「一番小さいタイプのコンピュータのCPU」，つまりそれまでのコンピュータから不必要な機能をはずしてぎりぎりまで小さくしたプロセッサとして開発し売り出したのです。

そしてこのインテルとモトローラの争いは80系＝日電，シャープ，ソニーetc.……/68系＝日立，富士通（最近富士通は80になっちゃったけど）etc.と日本のメーカーを巻き込んでの争いとなり日本でも一般のユーザーが80派と68派に分かれてしまいました。

そして，先発したためソフトなどの資産の多い80系ユーザーが「68なんかソフトが少ないくせに」といえば元がコンピュータから降りて来たアーキテクチャなどのエレガントな68系のユーザーが「やーい，電卓上がりの80にはこんなエレガントなプログラムはできんだろー」（昔はよくこれをやって人をいじめたもんだ）という醜い時代が続いたのです（ちなみにCPUがCPUと呼ばれるのは世間に80派の人が多かったせいなんですよ。だって68が多かったらMPU（マイクロプロセッシングユニット＝モトローラ系でのCPUの呼び方）になってたはずなんだから）。

そして，現在。先に開発して汚いアーキテクチャのCPUを乱発しユーザーの多いインテル系と綺麗なMPUを作って固定ファンも多いのにいつも後手後手にまわってユーザーが少なくなっちゃうモトローラ系という図式がいまだに続いているんですねー。うーん，恐ろしい話だ。

I/O

I/OがInput/Outputの省略形だということは既知のことでしょう。I/Oと聞くとディスプレイやキーボードがパッと頭に浮かんでくる人が多いと思いますが，それはこれらがパソコン本体とケーブルでつながって我々の目によく入るからでしょう。もっとミクロにCPUを中心に考えてみるとどうでしょう。たとえば，X1turboなんかではテキストVRAMをメモリに置かないで，I/Oに配置しています。これは我々の目には見えてませんが，パソコンの中ではちゃんとI/Oポートを介してCPUとテキストVRAMの間で入出力が行われているのです。

つまりパソコン本体と周辺機器の間の入出力装置だけがI/Oなのではなく，もっと細かくみてCPUからみた入出力装置（コントローラなど）もI/Oと呼ぶことがあります。

CPUのアーキテクチャを考えるとき，ソフトウェア面とハードウェア面の２つのアプローチがあります。もちろん，これら２つは互いに密着した関係にありますが，たいていは，どういう動作をさせるかという観点から，まずソフトウェア面での仕様を設定するようです。現状のハードウェアの進歩がめざましく，ほとんどの要求はハードウェア的に満足させることができ，性能を制限するのはプログラミングする側なのです。では，さっそくソフトウェア的アプローチから始めましょう。

CPUのソフトウェア的アーキテクチャを決める要素はほぼ次のとおりです。

1) データ形式
2) レジスタ構成
3) 命令セット
4) アドレッシングモード
5) 割り込み処理

これらのものをどういうふうに位置づけていくかでCPUの性格が決まります。以下に，ひとつずつ細かく見てみましょう。

データ形式

データの種類は，論理データ，数値データ，文字データに大別されます（図２）。

●論理データ

コンピュータは２進数の論理回路で構成されていますので，コンピュータの扱うデータも１（真）および０（偽）の論理データがもっとも自然なかたちといえます。

１か０かのデータをビットと呼び，通常は８個あるいは16個（最近では32や64もある）をまとめてひと組のデータとして扱っています。８ビットパソコンであるX1と16ビットパソコンであるX68000との基本的な違いは，CPUが同時に何ビットまとめて扱えるかの違いなのです。

●数値データ

数値データについても論理データの延長として考えることができます。というのは，数を数えるのも，ものがある（1）かない（0）かを調べていることになるからです。しかし，数が大きくなると１個ずつ数えるのは大変です。

そこで，古代の人は位取記数法というものを考え出しました。図３は２進法の仕組みを示した図です。２進法では各位を2^0，2^1，2^2，2^3というように２のベキ乗のまとまりとして，どの位があるかないかで合計の数を表しています。したがって，０，１，２，３，４という連続した整数も２進法を使えば，１と０だけで表せるわけです。そ

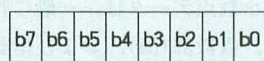
図２　いろいろなデータ形式

b0〜b7: ビットデータ(1or0)

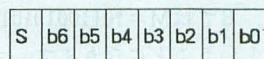
図2.1　論理データ

S: 符号ビット(正→0, 負→1)
b0〜b6: 2進データ(1or0)

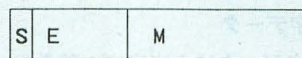
図2.2　固定小数点データ

S: 符号ビット(1ビット)
E: 指数部(2進データ)
M: 仮数部(2進データ)

図2.3　浮動小数点データ

図３　２進数の計算（１）

1 0 1 1 0 0 1 0
↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑
2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0 の位

$$2^7×1+2^5×1+2^4×1+2^1×1=178$$

図４　２進数の計算（２）

```
 01010010 =  82     補数の計算はすべて
 10101101           反転して１を加える
 ─────────
 10101110 = -82

 01010010 =  82     １の桁上がりを除けば
 10101110 = -82     82-82=0 に相当
+)
 ─────────
100000000 =   0

 01100000 =  96     １の桁上がりを除けば
 10101110 = -82     96-82=14 に相当
+)
 ─────────
 10101110 =  14
```

ビット

情報の最小単位を表します。最小の情報とは「はい」，「いいえ」で答えるもの，すなわち，ものが「あるか，ないか」，「真か偽」かといった情報です。

こういった真か偽かという「論理」を代数として扱えるようにしたのがブール代数です。

コンピュータが扱うのが得意とされている２進数も結局は論理として処理されています。つまり，

10011010

という２進数は「真偽偽真真偽真偽」という論理の集まりとして見なされ，論理回路で構成された演算器で数値としてつじつまがあうように演算されています。さらにいえば私たちがBASICなどで，

PRINT A+B

などとしただけでも，CPU内部では数えきれないくらいの「はい/いいえ」が飛びかっているわけです。

最近はファジー理論などでものごとは0/1では割り切れないということが強調されていますが，どちらかといえば，ものごとをすべて「はい/いいえ」で割り切れるようにとらえよう，という発想のほうが実は凄かったのではないかという気もします。

バス

CPUがデータを出し入れする信号線をバスといいますが，バスにはメモリのアドレスを指定するためのアドレスバスとデータをやり取りするデータバスがひと組ずつしかありません（ここでいうひと組とは，８ビットCPUならばデータバスは８本がひと組になっているということです。アドレスバスは８ビットCPUのZ80でも16本あります）。したがって，８ビットCPUで２つの16ビットデータの加減算を行うときは，まず片方のデータを８ビットずつ２回に分けて読み込んで保持しておき，続けて同様に２つ目のデータを読んでそれらの間で演算を行います。

CPUのビット数

CPUっていえばX68000に使われているMC68000は16ビット，X1やMZで使われているZ80は８ビットCPUなわけですが一般にCPUのビット数はCPUがメモリとデータのやり取りする「データバス」のビット数ということになっています。

つまりハード的な理由からきているわけですが，実際にマシン語でプログラムを組んでみるとこれらがソフト的な感覚とは本当に一致しない（これはデータバスのサイズをレジスタのビット数と合わせるのが設計上いちばん楽になるはずなのでふつうはデータバスのビット数と一致して当然なはずなのだ）ので実に奇妙な気分にさせてくれます。

CPUには計算に使うレジスタである「アキュムレータ」というものがあり，これの操作がマシン語プログラムの大半を占めるのでこれのビット数がそのCPUのソフト的な感覚となるわけなのですが……。

たとえば，IBM-PCで使われた8088は８ビットCPUですが，そのソフトは最新の80386を積んだPS/2でも同じように走ります。逆に68030のワークステーションで作ったプログラムでもハードやOSに依存していなければX68000でもほぼ動作しますし，メモリが足りれば68008で作ったような自作システムでも同じオブジェクトが動くはずです（68008は８ビットCPU）。

やがては実行速度をあげるため，64ビットや128ビットバスを持った内部32ビットのコンピュータも現れてくるでしょう（一度に命令を取り込むとメモリアクセス回数を減らすことができる）。そうなると……。

図5　レジスタの種類

ユーザーモード	データレジスタ アドレスレジスタ スタックポインタ フラグレジスタ
スーパーバイザモード	プログラムカウンタ システム制御レジスタ メモリ制御レジスタ 割り込みレジスタ

うすると，8ビットデータでは0から2^8-1（=255）までの整数データとして扱えるのです。

このままでは正の整数しか扱えないように見えますが，最上位ビットを符号ビットと設定することによって負の整数も表現できます。すなわち，最上位ビットが1のときには負，0のときには正と約束するのです。ここで，負の数に対しては，2の補数表現というものを使います(図4)。2進数には1の補数と2の補数の2つがあります。1の補数は元の数の0/1を反転したもので，元の数と足すとすべてのビットが1になるという性質があります。2の補数は1の補数に1を加えて作りますので，元の数と加えるとすべてのビットが0になり（つまり0になる)，負の数を表すのに都合のいい性質を持っています。この補数は減算を行うときに，加算と同じような計算ですむという利点があります。詳しくは図中の説明を見てください。

コンピュータは2進数の扱いが得意ですが，人間が通常使う数値は10進数なので，2進化10進数（Binary Coded Decimal code：BCD)というデータ形式をサポートしていることがあります。これは，10進数の1桁ごとに4ビット長の2進数を当てはめ，それを上の桁から順番に並べて表しています。たとえば，10進数で369という数は，BCDで001101101001となります。

実際のCPUで行う演算は，これまで述べた論理データか符号付き整数データに限るものがほとんどなので，アーキテクチャを考えるうえではこれまでの説明で十分です。しかし，コンピュータを扱ううえでは実数データと文字データの知識も重要なので，ここでも少し触れておきます。

実数データの表現方法には，固定小数点データと浮動小数点データの2通りがあります。固定小数点データのほうは小数点の位置を文字どおり固定させておくもので，整数部8ビット，小数部8ビットに設定することが多いようです。しかし，この方法では小さいほうの小数では，有効数字がまったく取れません。たとえば有効数字が16桁あるA=10110010.10111001という数を15桁小さくした数はB=00000000.0000000 1となって，有効数字がたったの1桁になってしまいます。

これに対して，有効数字は101100101011 1001のままにして，小数点の位置をずらしていくように表現したのが浮動小数点データです。これならばAは右に8桁（+8），Bは左に7桁（−7）ずらしたものと書けます。しかも有効数字の桁数は変わりません。浮動小数点データは，一般に符号用ビット(S)，仮数部(M)，および指数部(E)からなり，その値は$(-1)^S \times M \times r^E$で表されます。rは基数といい，ふつうは2をとります。AではM=1011001010111001，E=+8，BではM=1011001010111001，E=−7ということになります。コンピュータの高級言語を使って数値計算をする場合には，このような実数データの表現が問題になってきます。

●文字データ

文字データは1文字ずつ整数のコード番号と対応がつけられていて，正の整数データに変換して扱われます。皆さんには，ASCIIコードとしてお馴染みでしょう。また文字列の場合には，文字列の長さと文字コードの列とがセットでデータとなる場合があります。

*　*　*

このように，いろいろなデータ型があっても，CPUにとってはどれも単にビット列としてしか扱いません。たとえば，

01000001

というビット列は，数値なら65，文字なら“A”，前に，

00110100

というビット列があると“漢”という字の一部であることもあります。結局は利用者（またはプログラム）がそのときどきに応じて使い分けているのです。

レジスタ構成

レジスタはCPUの中にあって，データを一時的に保持しておく回路のことです。レジスタがメモリと違う点は，CPUの内部にあるためにCPU内の各部とダイレクトにつながっていて，アクセスが高速だということです。また，CPUの命令も各レジスタごとに用意されているので処理が一発でできるのです。一方メモリにアクセスするためには，まずメモリのアドレスを設定してからデータを読み出すという二度手間が必要なうえ，しかもそのデータを演算に使いたいときは，やはりいったんレジスタに保持させなければならないのです。

アーキテクチャとしてのレジスタ構成を考えるうえでは，レジスタの本数とその役割分担がポイントとなります。レジスタの長さそのものは，扱うデータの長さに合わせればよいわけで，さらにレジスタ長よりも長いデータを扱うときは複数のレジスタに並べるようにするのが一般的です。

レジスタの本数を多くすれば，より多くのデータをより高速に処理できるという点では確かに有利ですが，その代わりCPU内部のバスの接続など，回路が複雑になるうえに，CPUの命令もそれだけの数をサポートしなければならず，一概に有利ともいいきれません。したがって，多くの場合，各レジスタの役割を制限する手が使われます。

この制限が少なければ「直交性がよい」といわれ，ソフト開発者に歓迎されることになるでしょう。比較的美しいといわれるMC68000ぐらいになるとアドレスレジスタとデータレジスタの区別だけで，それぞれ同等なレジスタがそれぞれ8本ずつぐらい用意されていますが，PC-9801に使われている8086系列でさえ命令によって使えるレジスタがきっちり決まっているのです(たとえば，ループカウンタにはCXレジスタ，掛け算の掛けられる数にはAX，掛ける数にはBXレジスタなど)。

これはちょうどZ80では，演算はAレジスタ(アキュムレータ)，ループカウンタにBレジスタ，I/OアクセスのアドレスにはCレジスタと決まっているのと同じです。

そのほか，Z8000などは8ビットから64ビットまでかなり自由にレジスタを割り当てられたり，V60など32本の汎用32ビットレジスタという究極の直交性を誇るCPUもあります。

レジスタの一般的構成はメモリのアクセスアドレスを指定するアドレスレジスタ，

実際にデータを格納するデータレジスタのほか，プログラムカウンタ，スタックポインタ，割り込みベクトルレジスタ，フラグレジスタなどからなっています（図５）。

プログラムカウンタはメモリ上の命令を逐次読み出して処理するノイマン型に特徴的なもので，ひとつ命令を読み出すごとに１ずつ増えていき，次に読み出すべき命令のアドレスを記憶しておきます。もちろん分岐命令でジャンプしたときは，ジャンプ先のアドレスがロードされます。スタックポインタはスタックと呼ばれるデータ記憶領域に一時的に退避させられたデータの先頭のアドレスを記憶します。

フラグレジスタは，演算の結果の正・負，ゼロ・ノンゼロ，繰り上がり・繰り下がりなどを記録しておくところで，演算結果によるプログラムの分岐ではこれを見て判断します。ソフトウェアからアクセスできるレジスタは以上のとおりですが，実際にCPUの中にはハードウェア上の処理を進めるためにたくさんのレジスタがあります。

命令セット

CPUに対する命令は，動作の内容を指定するオペレーションコード（オペコード）とその動作が施されるデータのことをいうオペランドの組からなっています。アセンブラを知っている読者にはもう周知のことでしょう。

アセンブラのニーモニックで書けばLD，ADDなどといった命令もやはり８ビット（16ビット）の２進データの形でCPUに取り込まれるのです。たとえば，Z80ではLD A,Bという命令は８ビットコードで78_H，ADD A,Bは80_H，INC Aは$3C_H$というぐあいです。

これら３つの命令で代表されるように，オペランドがレジスタであればそのレジスタの指定はオペレーションコードに含まれますが，LD A,32Hというように直接の数値データ（イミディエイトデータ）のときには，オペレーションコードの次にデータが続いて命令となります。CPUには最初のオペレーションコードを解読したときに，それだけで実行できるのか，それとも次にどういう種類のデータがくるのかがわかっているのです。

では，具体的にどのような命令が取り上げられるか見ていきましょう。図６が一般的な命令の一覧です。これらの命令セットを大きく分けると，

a）　データ転送命令
b）　データ演算命令
c）　プログラム制御命令
d）　I/O制御命令
e）　CPU制御命令

となります。

●データ転送命令

データ転送命令は，メモリとレジスタの間，レジスタ同士の間，あるいはメモリ同士の間でデータを転送するものです。メモリ同士の間でのデータ転送では一度に大量のデータを移すことが多いので，ブロック転送命令というのもあります。これは，転送するデータの先頭アドレス，データ数，転送先のアドレスを指定すれば，一発でそのブロックを移してくれるものです。また，スタックへの一時退避，スタックからの復帰は標準的な命令で，たいていのCPUはサポートしています。

図６　一般的な命令セット

データ転送命令
　Z80：LD, PUSH, POP
　68000：MOVE, MOVEM
データ演算命令
　算術演算
　　Z80：ADD, SUB
　　68000：ADD, SUB, MULU, DIVU
　論理演算
　　Z80：AND, OR, XOR
　　68000：AND, OR, NOT, EOR
　シフト演算
　　Z80：SRL, SRA, SLA
　　68000：LSR, ASR, ASL, LSL
プログラム制御命令
　Z80：JP, JR, CALL, RET, NOP
　68000：JMP, JSR, BRA
I/O制御命令
　Z80：IN, OUT
CPU制御命令
　Z80：HALT, SCF
　68000：RESET, STOP

レジスタ

コンビニなんかでインスタントラーメンを買うときに「レジでお金を払う」っていうよね。あのレジっていうのは英語のレジスター(register)を語源としているんです。これは「記録」っていう意味があるんだけど，あのレジ(スター)がお金を保存する道具なら，CPUの中にも計算結果などを一時的に保存するレジスタが用意されています。コンピュータのレジスタにはユーザーが直接使うことのできるものと，CPU内部で使われてユーザーの使うことのできないレジスタがあります。

ニーモニック

我々がコンピュータに命令を与えたいとき，BASICやＣなどの高級言語ならば機械語を特に意識する必要もありませんが，ハードの機能を直接に使いたいときなどはどうしても機械語でプログラムを組む必要がでてきます。コンピュータではすべての情報を数値で扱うので，16進の数値を直接与える機械語はコンピュータにとってはいちばん理解しやすいものですが，我々にとってはただの数字の羅列にしか見えません。そこで我々が機械語を扱いやすいように，ニーモニック（アセンブリ言語）が考えられました。たとえばZ80でＡレジスタに１を代入する命令は16進数で直接書けば，

```
3E 01
```

ですが，ニーモニックであれば，

```
LD A, 1
```

と表されます。どちらが理解しやすいかは一目瞭然でしょう。そしてニーモニックをCPUが理解できるような数値に変換するプログラムをアセンブラといいます。余談ですがマシン語に精通してくるとダンプリストを見ただけでニーモニックが頭に浮かんで来てしまうそうです（うーん，しかしMC68000でこれができる人がいたら凄い）。

スタック

サブルーチンを呼び出したあとの戻り先などを記録する場所がスタックで，スタックポインタと呼ばれるレジスタがスタックの場所を示します。スタックは最後に入れたデータを最初に取り出すLast in first out方式（LIFO方式）です。

たとえば，我々が読み終わった本を順番に重ねていくと，常に最後に読んだ本がいちばん上になります。つぎに読み終わった本を取るときには山の途中の本を抜き出したりすると，山が崩れてしまいますから，いちばん上の本から順序よく取っていくことにします。コンピュータのスタック構造は，まさにこの本の山にたとえることができます。

割り込み処理

割り込みの具体例としては，プリンタとの通信，タイマ制御などがあります。プリンタの印字のとき，プリンタ本体の動作速度がきわめて遅いために印字中にCPUが動作を止めて待機するのは時間の無駄です。そこでプリンタはバッファメモリを持ち，印字データをそこにロードしておき，CPUはほかの処理を行っています。バッファメモリが空になってもまだ印字終了にならないときには，プリンタはCPUに印字データの送信を要求しますが，CPUはそのとき別の処理を行っているので，割り込みが必要となるわけです。また，ゲームなどでBGMをならすとき，演奏にはテンポを一定にキープしなければなりませんが，CPUは常にキー入力やキャラクタ移動など別の処理で忙しいのです。そこでタイマを作動させておき，一定間隔でBGMルーチンに処理を移すようにしています。これも割り込みの典型的な例です。

●データ演算命令

データの演算には，加減算のほか，AND・OR・NOTなどの論理演算，ビットのシフト・ローテートが一般的です。条件判断のための比較，ビットごとのセット・リセットも広い意味で演算といってもよいでしょう。8ビットCPUでは四則演算は加減算が限度で，乗除算はたいていシフトと加減算の組み合わせで処理していました。最近のCPUでは乗除算もサポートされていますが，これもCPUの内部でシフトと加減算を組み合わせているようです。

さて演算の結果，正・負，繰り上がり・繰り下がりなどに対してはフラグレジスタの中身を書き換えます。比較のときは，実際に演算結果は求めませんが，フラグだけは書き換えて条件判断に対応します。このフラグレジスタもビットごとのセット・リセットが可能です。

●プログラム制御命令

プログラム制御命令は，ジャンプ，条件分岐，サブルーチンコール・リターンのことです。本質的にはプログラムカウンタの内容を操作しています。ジャンプのときにはジャンプ先のアドレスをロードし，条件分岐では比較演算を行ったあとにフラグを調べて，その結果に応じてジャンプアドレスをロードします。サブルーチンをコールするときは，呼び出す元のアドレスをスタックに退避させておいてから処理をサブルーチンに移します。リターンがきたら，スタックから元の位置のアドレスを戻してくるのです。

図7　アドレス指定の方法

●I/O制御命令

I/O制御命令はアクセスするI/Oのアドレスを指定して，データを読み書きします。これにもブロックアクセス命令があります。ただし，メモリマップドI/Oの場合は，通常のデータ転送命令と変わりません。

●CPU制御命令

CPU制御命令は，割り込み処理あるいはもっと高度なCPUでサポートする例外処理に対応するものです。例外処理というのは，CPUがメモリに順番に格納されている命令を逐次実行している状態から，なんらかの外的要因によって特別の動作状態に移行することをいいます。これには，外部ハードウェアから突然送られてきた信号に応答してCPUの処理を一時的に移す「割り込み」のほか，エラーが生じたときに暴走を防ぐための処理などがあります。詳しくは「割り込み」のところで説明しますが，命令としてあるのは，割り込みの許可・不許可や例外処理をソフト的にシミュレートさせるものなどです。

＊　　＊　　＊

機械語プログラムの解説を読むと，外国語の文法書のようにいろいろな命令が次々と並んでいて，なんとも繁雑に見えます。機械語（アセンブラ）が難しく見えるのは命令の多さに圧倒されて手のつけようがないように思われるからでしょう。しかし，それらの命令は上の5つのグループのどれかに属しています。全体としてきちんと体系化されていることがわかれば，多少は理解できそうな気になってきませんか。

アドレッシングモード

命令セットのアーキテクチャを考えるうえでは，オペレーションコードの種類を決めることはもちろん重要ですが，オペランドにどういう形式のデータがとれるかという点もCPU設計のための大きなポイントとなります。

レジスタ構成のところで述べた，レジスタの役割が決まっている場合というのは，あるオペレーションコードに対してとれるオペランドが決まっているということになります。オペランドがレジスタの場合は，オペランドとしてとれるレジスタの種類の数だけオペレーションコードが必要になるので，それだけ命令体系も複雑になってしまうのです。

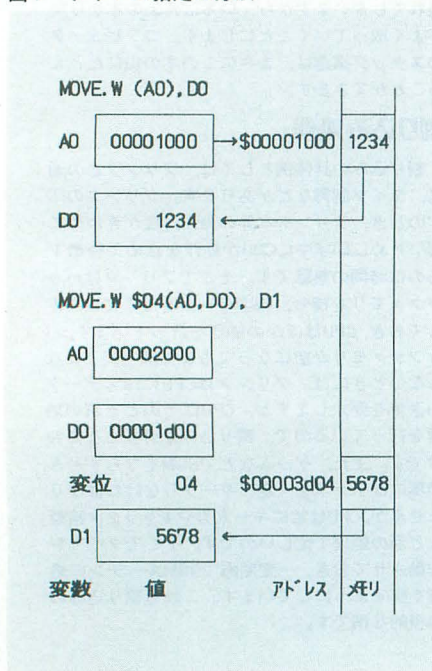
さらにLD A,(BC)のようにBCをアドレスレジスタのように使ってメモリのアドレスを格納しておき，間接的にそのメモリの中身を指定する方式もあります。このようなオペランドの指定の方法をアドレッシングモードといいます。このアドレッシングモードに従って命令セットもレジスタ構成も決まるといってよいほど，互いに密接な関係があります。

アドレスの指定の方法は，図7のように直接指定と間接指定，絶対指定と相対指定との組み合わせで計4種類に分類されますが，直接相対指定というのはまずありません。

直接絶対指定では，アドレスレジスタの中身がそのまま処理されます。MC68000のアセンブラではMOVE.W An ,Dnがその一例です(以下，サンプルは68000のアセンブラ)。これは，アドレスレジスタAnに格納されている値をデータレジスタDnに転送する命令です。

間接絶対指定では，アドレスレジスタが指定するアドレスのメモリに格納されている内容を処理するものです。文章ではわかりづらいと思いますので，実例に沿って図で説明します。MOVE.W (A0),D0という命令において，まずA0に$00001000が入っているので$001000番地の中身を読みにいきます。直接指定では$00001000がD0の内容になってしまうのが違う点です。そして，$001000番地の内容$1234がD0に転送されるのです。

間接相対指定は，アドレスレジスタの内容とオペランドの一部として与えたディスプレイスメントを加えた値をアドレスとして指定するものです。実例として，MOVE.W $04(A0,D0),D1という命令を図で説明します。A0に入っている$00002000にD0に入っている$00001D00とさらにオペランドにある$04を加えた合計$003D04をアドレスとするメモリの内容を読み出します。そしてその中身である$5678がD1に転送されるのです。

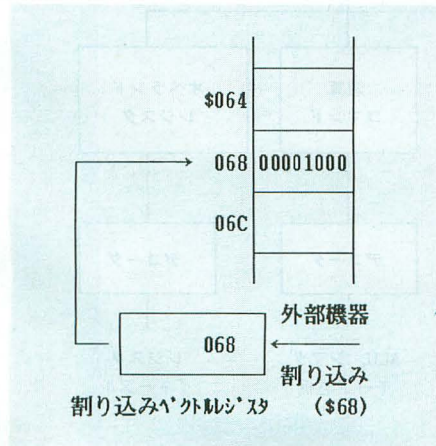
もっと原始的なアドレッシングモードと

してオペランドに数値を与えるイミディエイト指定というのがあります。たとえば，MOVE.W $1000 ,A0というのは，$1000番地の中身をA0に転送する命令で，MOVE.W #$1000 ,A0というのは$1000という値をそのままA0に転送するものです。

基本的には上の4つで十分で，確かにZ80にはこの4つのアドレッシングモードしかありません。しかし，MC68000を見るとプリデクリメント・ポストインクリメントアドレスレジスタ間接指定という超強力なアドレッシングモードがあります（Z80ではブロック転送命令でかろうじてサポートしている）。

たとえば，MOVE.W (A0)+,D0という命令ではA0が指定するアドレスの内容をD0に転送したあと，自動的にA0の値を2だけ（ここではワード＝2バイト転送したから）進めてくれるのです。ですから，MOVE.W (A0)+,(A1)+という命令を繰り返すだけで，A0を先頭とするブロックからA1を先頭とするブロックへと自動的に転送が可能となるのです。

割り込み処理

CPUはコンピュータ内部で生成されるクロックと同期して動作を進めます。しかしI/Oポートに接続されている外部機器はCPUのクロックと同期しているとは限りません。このとき外部機器とCPUとがデータをやりとりするには，外部機器から非同期的に通信を要求する信号が送られてきたときに，CPUは現在実行している処理を一時的に中断して，I/Oアクセスの処理をするルーチンに処理を移します。

図8　割り込み処理

実際には，外部機器から送られてきた割り込み要求信号はCPU内に保持され，ひとつの命令を実行するたびにその検出を行います。割り込みが検出されると，CPUは割り込みをかけた外部機器から割り込みベクトルというものを読み込みます。この割り込みベクトルは割り込み処理を行うルーチンのスタートアドレスを格納しているテーブルのアドレスを指定します(図8)。CPUは割り込みベクトルに従って処理ルーチンに実行を移します。このとき，それまで実行していたプログラムカウンタの値を記録しておき，割り込みルーチンが終わると元の状態に実行を戻します。このように，割り込みはハードウェアとソフトウェアの協力によって実現されます。

外部に割り込み要因がある場合のほか，ソフトウェア上のバグなどによりCPU内部でエラーが出たときにも復旧のために処理を移すことがあります。たとえば，MC68000では割り算命令で0で割ったとか，誤ってデータ領域にジャンプしてデータを命令として不当に読み込んだとかなどが当たります。これらはより広い意味で例外処理と呼ばれます。

また高度な処理として，特権違反というものがあるので，ここで簡単に説明しておきます。ある程度高級なCPUにはユーザーモードとスーパーバイザモードの2種類があります。ユーザーモードはユーザーが自分のプログラムを実行させるためのモードでいつもはこのモードになっています。

このほか，システム全体の動作を左右するSTOPだとかステータスレジスタの操作だとかいった命令はユーザーモードから誤って使用されると間違いなくシステムを混乱させます。そこで，こういった特殊な命令（特権命令）はスーパーバイザモードからしか使用できないようにしてシステムの安全性を高めているわけです（ちなみに，あのMacintoshはすべてスーパーバイザモードで動いているそうですが）。

通常のアプリケーションはユーザーモードで走らせるのが安全です。ユーザーモードにおいてこの特権命令を実行させようとすると，特権違反としてエラーとなりCPUは処理を移します。

逆に，例外処理では必ずスーパーバイザモードに入るので，ユーザーがスーパーバイザ領域にアクセスしたいときには，TRAP命令で故意に例外処理を起こすという技もあります。

以上，CPUのソフトウェア的アーキテクチャについて，駆け足で説明してきました。個々の説明が多少不足気味だったかもしれませんが，コンピュータがどういうコンセプトのもとで構成設計されているか，その雰囲気だけでも伝わったかと思います。それでは次に，いま述べたような命令を実現するようなCPUのハードウェアの仕組みに迫ってみることにしましょう。

3 CPUのハードウェア的アーキテクチャ

CPUのハードウェア内部は，そこで処理される命令，データを実際に保持・転送・演算するためのデータバス回路と，データバス回路の動作を制御するための制御回路とから構成されています。データバス回路は文字どおりデータの通り道で，アーキテクチャの中心は制御回路の構成にあるといってもよいでしょう。

ハードウェア的アーキテクチャを説明する前に，ハードウェアを構成する回路部品の基本単位を復習しておきます。個々の回路の動作について詳しくは本誌89年1月号の特集記事を参照してください。

さて，データを保持するレジスタには，Dフリップフロップが使われます。このDフリップフロップはクロックに同期して入力データを保持します。複数の入力からひとつを選択する回路にはデータセレクタを用います。データセレクタには，チャンネル分の入力端子とひとつの出力端子，それにチャンネル選択端子とがあり，チャンネル選択端子に選択するチャンネルを指定して出力からデータを取り出します。演算装置（ALU）は整数データを入力として加減算や論理演算を行い，その演算結果を出力します。

シフトレジスタは，シフト演算を行います。これは，ロードされたデータに対し1ビットずつシフトさせていくものです。またカウンタはプログラムカウンタに使い，ひとつ命令を実行するたびにカウントアップしていきます。ジャンプなどでプログラ

ムカウンタの中身を書き換えることもありますので，ロード機能のあるカウンタが必要です。これらの基本回路の組み合わせでハードウェアが実現されるのです。

CPUは大きく分けると次の3つのブロックから構成されています（図9）。

1）命令読み出しとデコード
2）レジスタ内のデータに対する演算
3）メモリに対するデータのロード・ストア

命令読み出しとデコード

このブロックはプログラムカウンタと命令レジスタおよびデコーダからなっています(図10)。プログラムカウンタはパイプラインという多段処理に対応するために複数のカウンタが直列につながっています（パイプラインについてはあとで説明します)。

また，割り込み処理のときに割り込みベクトルを保持するレジスタ，現在のプログラムカウンタの内容を退避させておくレジスタもあります。

さて，プログラムカウンタに従って読み込んだ（フェッチされた）命令は，命令レジスタに保持されデコード回路によって解釈されます。実際，デコーダでは命令コードの各ビットの1/0の組み合わせから，ほかの各部分の回路のON/OFFを制御する信号を作り出します。

基本的には，デコーダはAND・OR・NOTの組み合わせ回路で，命令コードの各ビットを論理計算して，その結果ごとに異なる出力端子に信号を出すものです。それぞれの出力端子は，CPUの各部分に配線され，ここから出力された信号をもとに各部の制御回路が動作するわけです。逆に考えれば，CPUの各部分の制御回路はデコーダからの制御線を見て，ON信号がきたら自分の役割を実行するのです。

機械語の解説書を見ると，各命令ごとにオブジェクトコードのフォーマットが記されているのに気づくでしょう。たとえば，MC68000のMOVE命令では，（図11）第15,14ビットが00で，第13，12ビットが00でないということで，MOVE命令ということが決まります。さらに，第13，12ビットは転送データのサイズを示しています。第11～6ビットの組み合わせで第1オペランドの種類，第5～0ビットの組み合わせで第2オペランドの種類を表しています。命令コードはこの例のように，いくつかのビットごとにまとまって意味を持ち，そのまとまりごとにデコードされていくのです。

このように，機械語の命令コードは16進数で見るとよくわからないのですが，各ビットごとにその組み合わせとして見ると，意外と簡単に解釈できるものなのです。

レジスタ内のデータに対する演算

このブロックは，レジスタとデータセレクタの組み合わせでできているといえます(図12)。デコーダで解釈された命令に従って演算するとき，演算データはレジスタセットに格納されています。ですから，まずレジスタセットを用意しなければなりません。そして，そのオペランドに指定されたレジスタの出力を切り替えてALUあるいはシフタに入力してやります。ALUの入力は2つあるので，それぞれの入り口に，第1，第2オペランド用データセレクタとレジスタが設けてあります。また，ALUの出口にも演算結果レジスタがあります。ここで一旦結果を保持してから，オペランドに指定されたレジスタに書き込みます。

さて，入力が複数あるときには，どのレジスタから読み込むかはデータセレクタで選択しますが，出力が複数あるときは，すべてのレジスタに並列に出力してもかまいません。というのも，レジスタがデータを保持するには，クロックの立ち上がりが必要なので，書き込むべきレジスタのクロックのみ入るようにデコードしておけば，クロックの入らないほかのレジスタの内容には影響ないからです。しかし，実際にはレジスタの出力もON・OFFがあり，データを出力すべきレジスタを選択して，その出力だけONにするようになっています。

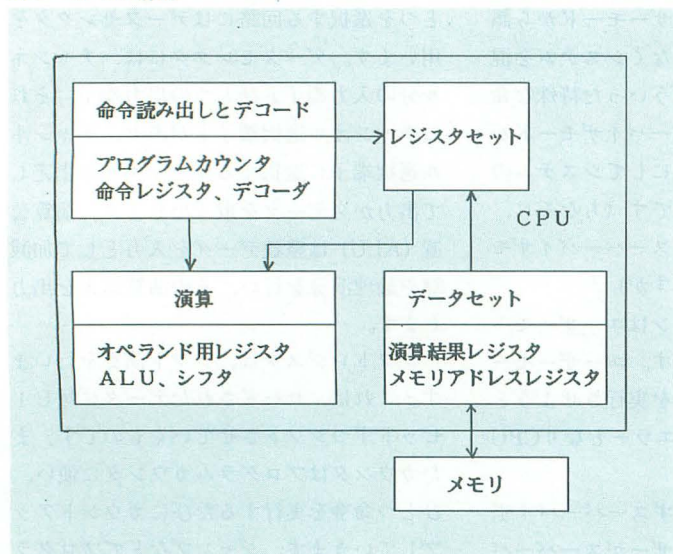
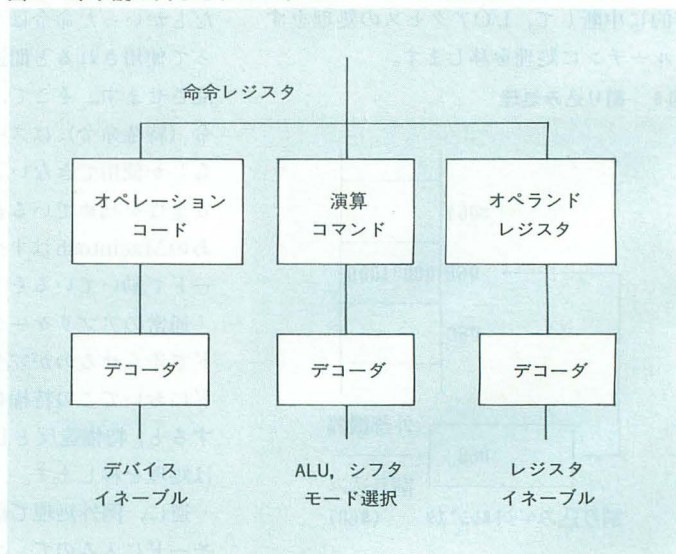
演算した結果によって条件分岐をするときなどはALUのキャリー・ボロー出力を命令コードの一部からデコードされた条件と比較して判断します。このためにALU出力を保持する条件レジスタとその比較を行うコンパレータとが用意されています。

メモリに対するデータのロード・ストア

メモリへのアクセスにも，アクセスする

図9　CPUの構成

図10　命令読み出しとデコード

命令レジスタ
オペレーションコード｜演算コマンド｜オペランドレジスタ
デコーダ｜デコーダ｜デコーダ
デバイスイネーブル｜ALU，シフタモード選択｜レジスタイネーブル

アドレスを保持するレジスタと読み込んだデータを保持するレジスタとを用意しておきます。演算結果のレジスタへの書き込みとメモリからI/Oへの書き込みとは同じレベルと考えて，演算結果レジスタとメモリ入出力レジスタとは共用のことがあります。

　ところで，命令コードとデータとはどちらもメモリから読み込んでくるのに，保持するのが命令レジスタであったり，メモリ入出力レジスタであったりするのは，どこで区別するのかと疑問に思うかもしれません。それは，最初にひとつの命令を読み込んで命令をデコードしたときに，そのあといくつオペランドとしてデータが続くかを判断し，読み込み先を振り分けるのです。

　ですから，プログラムミスなどからデータが不当に命令コードとして読み込まれてしまうこともあるのです。こうなるといわゆる暴走の状態になるわけですが，このとき本来命令にないコードが読み込まれると，先ほど述べたように，MC68000などのCPUでは例外処理に移って暴走を最小限に止めようとするのです。

＊　＊　＊

　以上，回路構成を簡単に説明しましたが，少々わかりづらかったかもしれません。そこで，実際の命令を例にとり，具体的なプログラムでデータの流れをシミュレートしながら，各回路の働きを追ってみることにします。シミュレートする命令は，以下のとおりです。

```
$001000:  207C00002000
          MOVEA.L #$002000,A0
$001006:  303C1234
          MOVE.W #$1234,D0
$00100A:  D050
          ADD.W (A0),D0
```

　まず，プログラムカウンタに$001000がロードされてスタートしました。命令コード$207C(＝0010 000001 111100)が命令レジスタに読み込まれます。これをデコードして，アドレスレジスタA0にイミディエイトデータを転送する命令と解釈しました。

　先頭の0010がMOVEA.L，次の000001がデスティネーションA0，最後の111100がソースイミディエイトを示します。次にくるデータはイミディエイトデータなので，メモリ入出力レジスタに切り替え，そこに#$002000を保持します。

　そして，メモリ入出力レジスタの出力をA0レジスタの入力に切り替え，このデータをA0に格納します。ここまでで1命令終了で，全部で6バイト命令だったので，プログラムカウンタの値は6だけ進んでいます。

　1命令終わっているので，次に読み込むデータは命令レジスタに入ります。命令コード$303C(＝0011 000000 111100)をデコードすると，先頭の0011がMOVE.W，次の000000がデスティネーションD0，最後の111100がソースイミディエイトを示し，これはイミディエイトデータをデータレジスタD0に格納する命令と解釈されます。

　次にくるデータもイミディエイトデータなので，メモリ入出力レジスタに切り替え，そこに$1234を保持します。そして，メモリ入出力レジスタの出力を今度はD0レジスタの入力に切り替え，このデータをD0に格納します。これで4バイト命令が無事に終了し，プログラムカウンタは4つ分進んでいることいなります。

　次にくる命令コード$D050(＝1101 000 001 010000)は，先頭の1101がADD，次の000がデスティネーションD0，そして001がワードデータの加算（.W)，最後の010000がソースにA0レジスタ間接指定がくることを示し，これはデータレジスタD0の内容にA0で指定されるアドレスの中身を加えて，その結果をD0に格納する命令と解釈されます。この命令では，第1オペランドがレジスタ間接指定なので，メモリアドレスレジスタにA0の内容を転送します。

　そして，A0の指定する内容を第1オペランド用レジスタに読み込んできます。次に，D0レジスタの内容を第2オペランド用レジスタに読み込みます。この2つのレジス

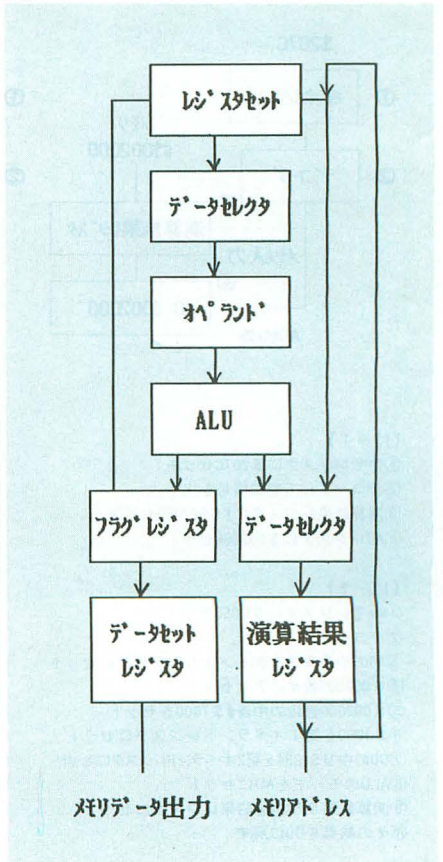



図11　機械語フォーマット

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	サイズ		デスティネーション実効アドレス レジスタ			モード			ソース実効アドレス モード			レジスタ		

デスティネーション実効アドレス(データ可変モード)

アドレッシングモード	対応ビット 11	10	9	8	7	6
Dn	レジスタ番号			0	0	0
(An)	レジスタ番号			0	1	0
(An)＋	レジスタ番号			0	1	1
－(An)	レジスタ番号			1	0	0
d16(An)	レジスタ番号			1	0	1
d8(An, IX)	レジスタ番号			1	1	0
Abs.W	0	0	0	1	1	1
Abs.L	0	0	1	1	1	1

ソース実効アドレス(すべてのモード)

アドレッシングモード	対応ビット 5	4	3	2	1	0
Dn	0	0	0	レジスタ番号		
An*	0	0	1	レジスタ番号		
(An)	0	1	0	レジスタ番号		
(An)＋	0	1	1	レジスタ番号		
－(An)	1	0	0	レジスタ番号		
d16(An)	1	0	1	レジスタ番号		
d8(An, IX)	1	1	0	レジスタ番号		
AbS.W	1	1	1	0	0	0
AbS.L	1	1	1	0	0	1
d16(PC)	1	1	1	0	1	0
d8(PC, IX)	1	1	1	0	1	1
# Imm	1	1	1	1	0	0

＊バイト・サイズ不可

サイズ	対応ビット 13	12
バイト	0	1
ワード	1	1
ロングワード	1	0

注）デスティネーション実効アドレス部は，レジスタ，モードの順でマップされている。

図12

タは ALU の入力に直結しているので，ALUをADDモードにするとその結果が即座にALUの出力に出てきます。その結果は演算結果レジスタに保持されますが，この命令は最終的に第２オペランドであるD0レジスタに格納される約束なので，演算結果レジスタの内容はD0に転送されて終わります。これで，プログラムカウンタは$00100Cまできているので，次の命令へと移っていくのです。

パイプライン処理

では，最後にパイプライン処理について説明してこの解説を終わることにしましょう。いま，CPUを機能別に３つのブロックに分けました。上のシミュレーションにおいて，これらのブロックをデータが通過していく様子を図13に書いてみました。

すると，CPUの動作は各ブロックごとにまとまっていて，そのなかでひととおり完結してから次のブロックに移っていくといってよさそうです。ということは，ひとつの命令を読み込んでから実行が終わるまで，待っている必要がないということなのです。つまり，最初の命令(MOVEA.L #$002000, A0)の命令のデコードが終わって，オペランドデータのセットが始まったら，次の命令（MOVE.W #$1234,D0）のデコードを始めてしまうのです。このような，ブロックごとに動作をずらしながらつなげていく処理をパイプラインといいます。

このように処理を並列して実行するわけですが，パイプラインの段数を多くするようにすれば，RISCプロセッサでなくても理論的には１命令＝１クロックまで実行速度を上げることができます。

ところでいま，動作が各ブロックで完結してから次に移ると述べましたが，プログラムカウンタと，オペランドレジスタの指定は１命令の間すべてのブロックに残っていなければなりません。というのは，前の命令でオペランドデータのセットを行っているときに次の命令のデコードをしてしまったら，アクセスすべきレジスタ(メモリ)の指定が書き換わってしまうからです。

したがって，パイプラインを実現するためには，イミディエイトデータの指定のためにプログラムカウンタと，命令コードの保留のために命令レジスタ（実際はオペランドレジスタの指定の部分で十分）とはブロック数分だけ用意しています。そして，パイプラインが１ブロック進むごとにそれらの中身もひとつずつ進んでいきます。ところが，現実問題として条件分岐などがあれば，すべての命令がすべてのブロックを順序よく進んでいくとは限りません。

また，前の命令の演算結果を次の命令のオペランドに使う場合などは，順序よく進んでいると間に合わなかったりします。最近のCPUでは，どうやってこのパイプラインの流れを乱さないようにするかという点で死力が尽くされています。このような難しい問題については，ここではこれ以上述べませんので，興味ある人はこのあとの中森氏の記事や参考文献を見てください。

図13　パイプライン処理

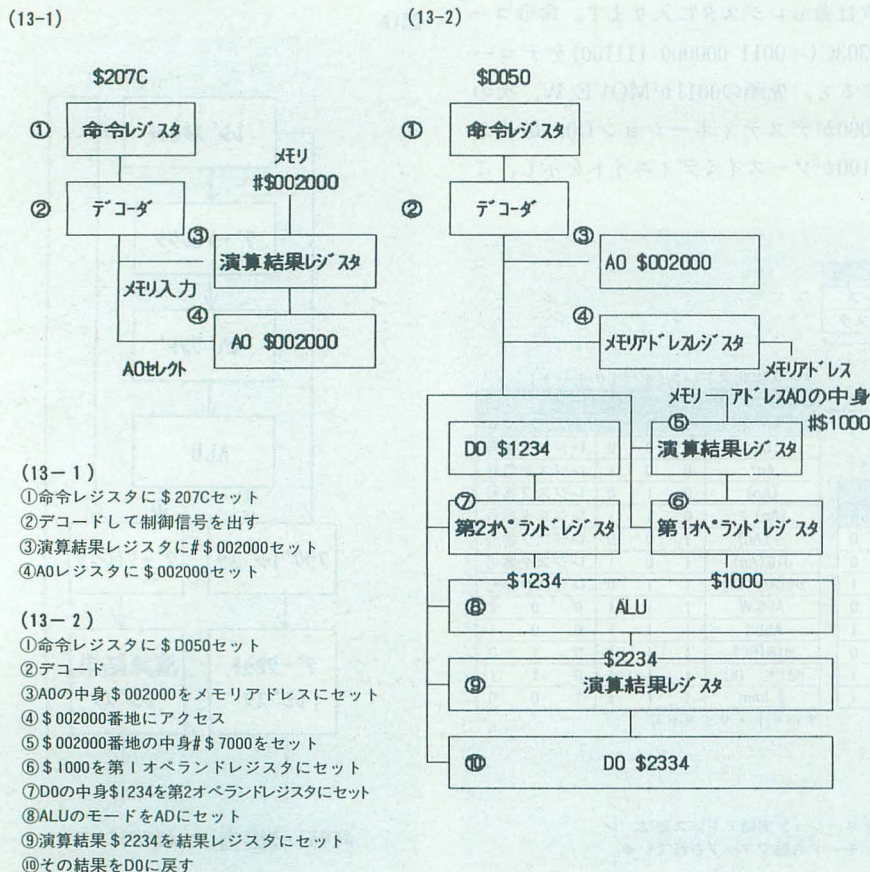
まとめ

この解説記事では，コンピュータの動作の中心となるCPUのアーキテクチャを考えながら，コンピュータとは基本的にどういうものかについてなるべく総合的に触れてみたつもりです。今回述べたアーキテクチャは，コンピュータのたどってきた発展の歴史を通じて，目まぐるしい技術や思想の革新に対しても最低限貫かれてきたノイマン型によるものです。

しかし，コンピュータシステムが大規模化，複雑化するにつれ，CPUとその命令セットのみで構成されたアーキテクチャでは不十分といえましょう。数理科学に基づく情報処理論，人間工学によるヒューマンインタフェイス論，最新テクノロジーのもたらす高度なハードウェアなどなど，コンピュータアーキテクチャに関連する分野は日に日に拡大しています。

コンピュータユーザーのひとりである私たちは，こうした技術の進化に押し流されて，逆に機械に使われてしまうという事態は避けたいものです。

◆参考文献
楠菊信，武末勝，脇村慶明：コンピュータの論理構成とアーキテクチャ，コロナ社（1988）
村岡洋一：コンピュータ・アーキテクチャ，近代科学社（1985）
宍倉幸則：68000プログラマーズ・ハンドブック，技術評論社（1986）

史上最低のCPU

RISCプロセッサの設計と製作

CPUの概略がわかったところで，さっそくCPUを作ってみましょう。なお，予告では「完全8ビット」となっていましたが，都合により「一部16ビット」に仕様変更されました。がんばれ日の丸CPU，目指すは史上最低のRISCプロセッサです。

Shimada Atsushi
島田 淳史

コンピュータの仕組みを覚えるには実際に触ってみるのがいちばんです。プログラミングを覚えるのも，実際に自分でプログラムを組んで試行錯誤するのが早道です。ハードウェアを理解するのも，自分で工作した経験の積み重ねでその仕組みを学んでいくのです。私の場合も最初にハンダゴテを握ったのは中学生の頃で，その当時は，いわゆるマイコンはまだワンボードマイコンのキット（TK-80だったかな）が出始めたばかりでした。私はもっぱらアマチュア無線のほうに熱中していて，通信機やアンテナなどの組み立てをしていました。しかし，経験も知識もなく，ほとんど失敗作ばかり。もちろん市販のものを買えばこと足りるわけだし，たとえ完成したとしてもコストパフォーマンスははるかに及ばないものしかできません。

それでも自作派を目指したのは，既製品を買ってしまうと，外側のケースをなでてやるぐらいしか可愛がる術がないからです。自分で苦心して組み上げたモノは隅から隅まで目が届いていて（部品の配置を間違えて，配線がスパゲッティのように絡まっているなんてことも知っている。もしかすると市販品の中でもそんなことがあるかもしれない……そんなわけないか），しかもどのスイッチを押すとどの部品が動作しているかなどといった仕組みもわかります。たとえ完動しなくても，1回失敗したところは次はなんとかうまく動作させようと頭を絞れば，それだけ身についていくことも多いのです。

電子回路にはアナログとデジタルとの2種類がありますが，経験からいってアナログ回路よりデジタル回路のほうが数倍も簡単です。正直なところ，私も無線機などの高周波アナログ回路では満足に作動させたものはありませんでした。しかし，モールス符号を打つためのエレクトロニクスキーヤーをC-MOSデジタルICで組んだときは，一発で動いたのです。デジタル回路の工作がこんなにもやさしいのかと感激したものです。実際ICの中身がどうなっているのか，まったく知らなくてもデジタル回路は組めるのです。

デジタル回路の基礎は本誌89年1月号でていねいに説明され，そこでは，我々ユーザーがブラックボックスとして使っているICは，ふたを開ければ基本的にAND，OR，NOTの組み合わせが機能ごとにパッケージの中に納められているだけのことだとわかりました。そして今月号では，デジタル回路の粋を集めたともいえるCPUの構成についての解説があり，やはりCPUといえどもふたを開ければ，いくつかのブロックの組み合わせであると述べられています。

結局，

1) ハードの仕組みを理解するには，実際に組み立てるのがいちばん
2) デジタル回路は工作も簡単
3) デジタルICの中身はいくつかの基本回路の組み合わせ
4) CPUも基本ブロックの組み合わせ

という4点を合わせると，「それではCPUの中身をバラして実際に作ってみよう」という結論になるわけです。

しかも，ここでは市販のTTLロジック回路だけで組めるものを設計します。TTL回路は秋葉原に行けば，1個30円ぐらいから高くても300円程度で手に入り，もう身近なものになったといってもよいのではないでしょうか。この記事でも代表的なTTL回路の働きを復習しながら，設計と製作の方針を説明するつもりです。

CPUを設計する

我々が自作するCPUはどんなに頑張っても280円で手に入るZ80よりよいものになるはずはありません。しかし，自ら設計してみることで，CPUがより身近に感じられるようになるだけでも十分でしょう。

さて，ここで設計するCPUは基本的な機能しか持たせていません。当然，ワイヤードロジックですからRISC（Reduced Instruction Set Computer）と呼んでいいかもしれません。すなわち，ある特定の用途のために，命令セットやアドレッシングモードを必要最小限に限って，その分製造コストの削減（ここでは製作労力や設計ミスの削減）を目的とするCPUのことです（本当は高速化が主目的なのでしょうが）。

ハードウェア的に制限が加わっているだけに，なにか複雑な処理をさせようとするとプログラムがたいへん大きくなってしまいます。まあ，今回は学習目的で実用性を問わないので，こういったRISCもどきで十分でしょう。ちなみに多くの命令を持つ高機能CPUはCISC（Complex Instruction Set Computer）といいます。まともに作ろうとしたら部品量が何倍にもふくれあがります。

まずはCPUのアーキテクチャの設定からです。先ほどの解説にもあるとおりアーキテクチャにはソフトウェア的アプローチとハードウェア的アプローチとがあります。さっそくソフトウェア的アーキテクチャから考えてみましょう。

●データ形式

TTL回路は，4ビットまたは8ビットまとめて処理するパッケージが多くあります。たとえば，1または0のデータを記憶するDフリップフロップにしても8個で1パッケージのものがあり，8ビットまでなら部品点数をかなりおさえることができます。ということで，データは8ビット整数ということでよさそうです。やはり，最低限とはいえ，8ビットはほしいところでしょう。ですからこれから製作するCPUは8ビットRISCプロセッサということになります。そして，条件分岐をするためには加算だけでなく減算も必要なので，データはやはり符号つき整数としておきます。

●レジスタセット

このCPUはRISCですから，レジスタも無理に汎用にする必要はなく，各レジスタの役割を明確に分けることにします。レジスタは，プログラムカウンタ1個，オペラ

ンド用レジスタが2個，演算結果レジスタ1個があれば，最低限こと足ります。あと，メモリアクセスにレジスタ間接指定のアドレッシングモードを設けるとアドレスレジスタが必要になりますが，以下で述べるとおりメモリアクセスはかなり高機能なMMU(?)に任せることになるので，これは演算結果レジスタと共用にします。

●**命令セット**

基本的なデータ転送，四則演算のみをサポートします。命令セットの一覧を図1に示します。命令の実行にはもちろんパイプラインなどなく，しかも簡略化のためすべての命令を1ステップで終わらせることにします。1ステップというのは，クロックが1個入ったらその命令は完了ということで，いい換えればすべてが1バイト命令というわけです。1命令＝1クロックというのはRISCプロセッサの基本です。

このままでは，命令コードの後ろに本来ならイミディエイトデータがくるような命令はとれないことになって不便なので，命令コードとイミディエイトデータとは別々のバスにします。命令とデータを分離して効率化を目指していますので，68040で話題の「ハーバードアーキテクチャ」と同じ観点に立っているといってしまいます。

これなら，イミディエイトデータの続く2バイト命令は，データバスにあらかじめデータを乗せてから，命令コードを命令バスにセットしてクロックを1個受ければよいのです。

しかし，このやり方ではバスは全部で16本あり，これを8ビットと呼ぶのはインチキともいわれそうですが，まあ，RISCということで今回は見過ごしてください(RISCのくせに16ビットというのはなんとも無駄な設計だなあ)。

しかも，命令を実行するためのクロックは人間が作るのです！　どうするのかというと，クロック発生の押しボタンがあり，命令コード（とイミディエイトデータ）をセットしたら，自分で押しボタンを押して1ステップ完了となるわけです。クロックといっても，等しい時間間隔で規則正しく1/0を送らなければならないわけではなくて，ひと組の1/0を1回送ると1クロックというのです。

このへんで嫌な予感がする人は，コンピュータを扱うセンスが優れている人です。それでは，命令セットはどのようにCPUにセットするのか？　もちろんメモリから読み込んで……なんて安心していると期待を裏切られます。なんとこのRISCでは，メモリ制御は人間がその代わりを務めるのです。だって，今回はあくまでもCPUの設計製作なのだから。メモリ（というよりメモリマネジメントユニット）である人間は（別に本当に頭の中にメモライズしていなくても，紙かなにかに記録しておけばよいのだ)，プログラムカウンタの中身を見て，そこに指定されているアドレスの命令をバスにセットします。プログラムカウンタの中身を見るのはLEDの表示から，そしてバスにセットするのはトグルスイッチから行います。

あと，もうひとつ人間に任せたいのは，分岐のための条件判断です。本来ならばフラグレジスタがあってその状態をCPU自身が読み取って条件判断を行うのですが，このRISCでは，フラグはLEDを並べて，フラグが立てばそのLEDが点灯する仕組みになっています。このフラグに従ってメモリからプログラムカウンタに分岐先のアドレスを読み込むわけですが，メモリも人間なので，実際には人間がフラグLEDを見て自ら分岐先のアドレスをセットすることになります。

●**アドレッシングモード**

アドレッシングはレジスタ直接絶対指定と間接絶対指定とをとるようにします。直接絶対指定はレジスタとレジスタの間の転送，演算ですから，命令コードの中で指定してダイレクトに実行できます。問題は間接絶対指定ですが，これはアドレス指定レジスタに読み書きすべきアドレスをセットしたのちに，それをメモリ（実際は人間）が参照して，データをやり取りするのです。メモリから読み込む場合はイミディエイトデータと同じようにデータバスにメモリ内の値をセットしてクロックを送り，メモリに書き出す場合にはレジスタ出力のLEDを見て書き取ります。

●**割り込み処理**

割り込み処理は非常に面倒なのでいっさいなし。

以上述べたように，ここではメモリアクセスのようなCPUの外部とのやり取りはすべてMMUの機能とし，人間自身がそれをエミュレートするという究極のアーキテクチャを採用しましたが，レジスタ間のやり取りのようなCPU内部の動作はほぼ実在のCPUのアーキテクチャに従っています。

ワイヤードロジックで1命令＝1クロックを実現していますから，理論的には仮にクロックを10MHzとすると10MIPS，33MHzなら33MIPSの性能が期待できますが，残念ながらメモリが追いつかない場合が多いようですね。

また，アドレスバスが8ビットなので，256バイトまでの物理アドレス空間しかアクセスできませんが，今回採用したMMUは(性能はともかく)非常に高機能であることが期待されるので，MMU側の仮想記憶機能を用いることにより，無制限の論理アドレス空間をアクセスすることも不可能ではありません。

冗談はさておき，それでは，皆さんお待ちかねのハードウェア設計編に移りましょう。

TTLロジック回路とその他の部品

まず部品を選択して，それぞれの部品の基本的な動作をざっと復習しておきましょう。以下の説明は設計した回路図（図2）を見ながら読んでもらったほうがよいかもしれませんが，全体の構成は後でまとめて説明しますので，今は個々の部品の使われ方に着目してください。

1）レジスタとフリップフロップ

データの一時記憶に使われるレジスタにはDフリップフロップ(D-FF)を使っています。Dフリップフロップは図3のようにNANDゲートを組み合わせてできているものですが，CK端子の入力がLからHに変わる（立ち上がる）瞬間のD入力のデータが保持され，その保持されたデータがQ出力から出力されます。

すなわち，1個のクロックの立ち上がりから次のクロックの立ち上がりまで，データを記憶しているメモリの働きをするわけです。ですから，これを8個並べれば，8ビットのレジスタとして使えるのです。D-FFのことをそのままレジスタと呼ぶことも多くあります。さて，実際には8ビットレ

図1　命令一覧

命令	内容
LD	レジスタ間，レジスタ-メモリ間のデータ転送
ADD	レジスタ間の加算
SUB	レジスタ間の減算
AND	論理和
OR	論理積
XOR	排他的論理和
SL	左シフト　CY ← [b7 ← b0] ←
SLA	算術的右シフト　CY ← [b7 → b0]（b7は自身に戻る）
SRL	論理的右シフト　CY ← [b7 → b0]
JP	ジャンプ（プログラムカウンタへのLD）

ジスタとしてLS273が手頃に使えます。しかし，ここではLS374を使います。これは3ステート出力というものを持っていて，コンピュータのバスラインを構成する回路を組むときには不可欠な性質ですが，詳しくはのちほど説明します。

2)　カウンタ

1クロックあるたびにカウントを進めるのに文字どおりカウンタというパッケージがあります。カウンタは図4に示すように基本的にはD-FFを直列に並べたもので，CKに1個クロックが入るたびにそれまでのカウント数を2進数で出力します。これがそのまま1命令終了するごとに命令の格納されているアドレスを先に進めるプログラムカウンタに使えます。

実際には，8ビットをひとつのパッケージに納めたものはなく，4ビットカウンタのみです。そこで8ビットカウンタとするには，4ビットカウンタを2個直列につないで使います。カウンタの最上位ビットを桁上がり信号としてそのまま次の最下位ビットに入れてやればOKです。

分岐命令があるときにはプログラムカウンタの中身をそっくりロードしなおすことになるので，使用するカウンタも0，1，2……と順番にカウントしていくだけでは駄目で，任意のデータをロードしてそこからカウントできるようなものにしておかなければなりません。このような機能を持つものをプリセッタブルカウンタといい，これならプリセットの入力端子をバスラインにつないでおけばよいのです。4ビットプリセッタブルカウンタのパッケージはたくさんありますが，3ステート出力を選ぶとすると，LS561しかありません。

3)　シフトレジスタ

シフトレジスタも図5に示すようにD-FFを組み合わせてできていて，これは8ビットデータを1ビットずつシフトさせるものです。このシフトの意味を算術として考えてみると，たとえば0011Hは3で，これを左に1ビットシフトさせた0110Hは6，さらに1ビットシフトさせた1100Hは12を表し，結局左に1ビットシフトさせるのは，その数値を2倍するのと同じことだとわかります。

図3　LS74

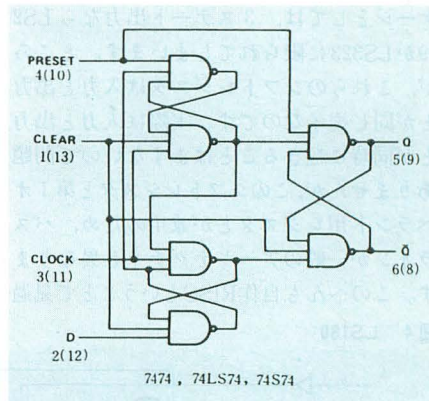
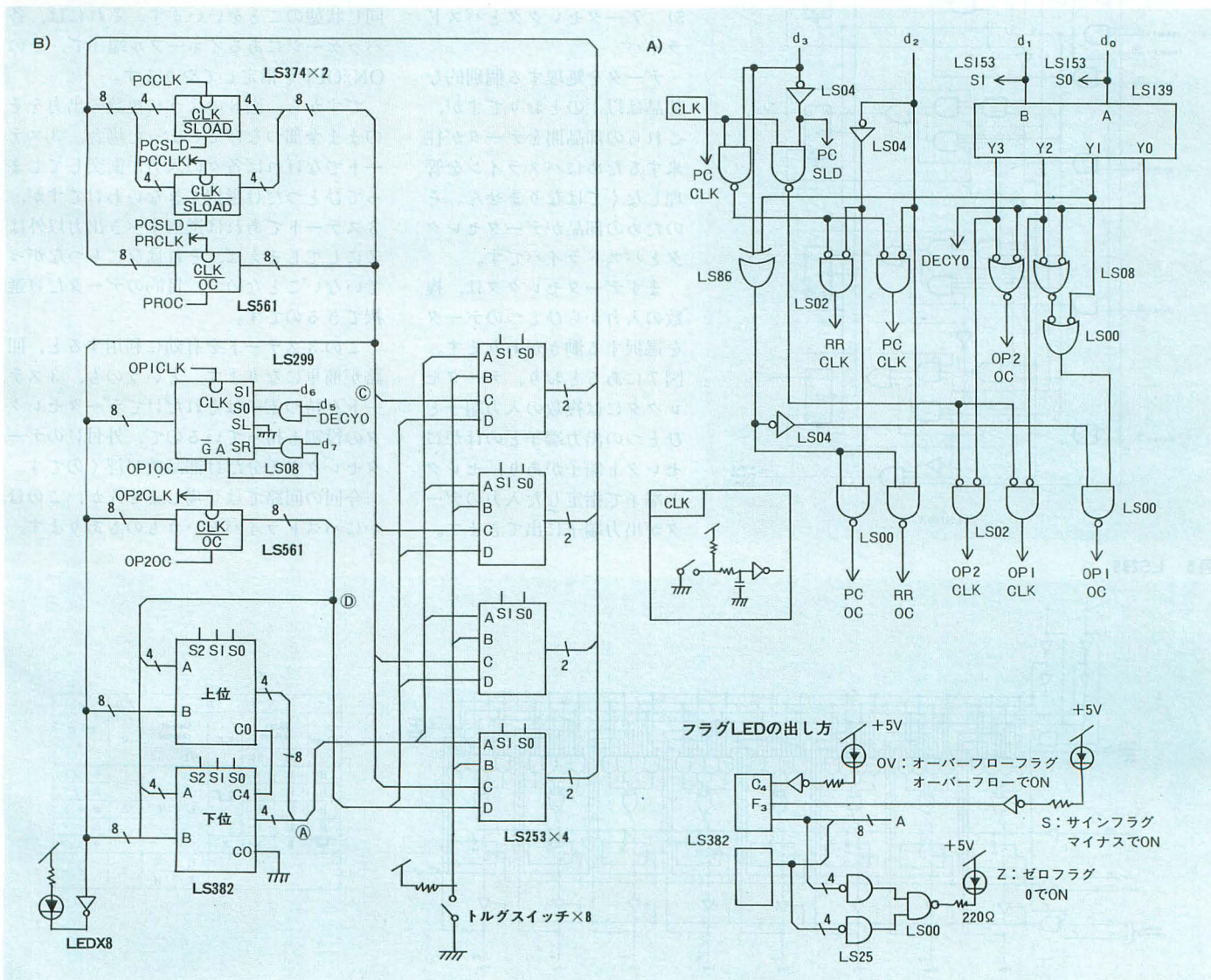
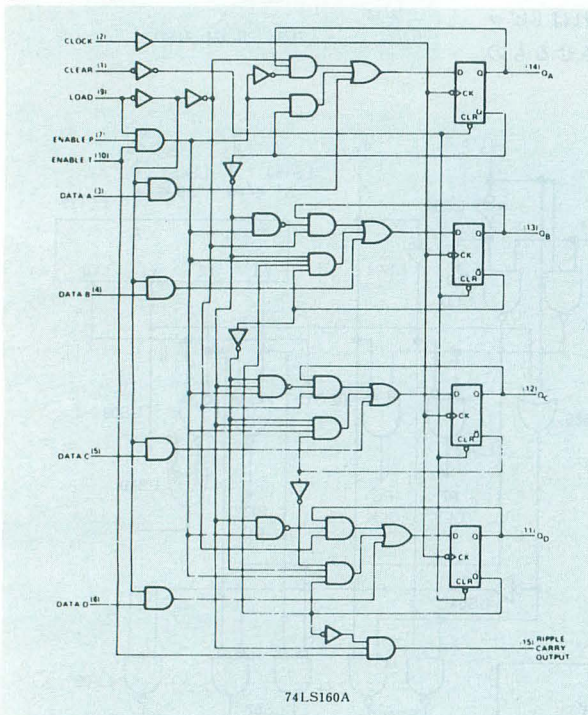
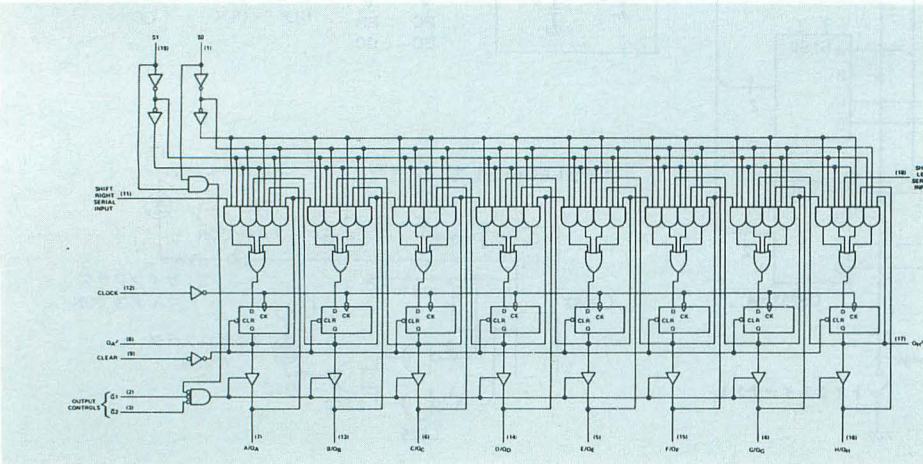



図2　回路図

同様に右に1ビットシフトさせるのは1/2にすることです。このようにシフトレジスタは乗算器として働くのです。さてこのシフトレジスタはビットシフトさせなければただのレジスタとして機能しますから，第1オペランド用レジスタと兼用とします。必要ならば，シフト操作のあとは演算結果レジスタに移してやることにします。パッケージとしては，3ステート出力ならLS299かLS323に限られてしまいます。ところが，これらのシフトレジスタは入力と出力とが同じ端子なのです。実際は入力と出力とが同時に起きることはまずないので問題ありませんが，このシフトレジスタと第1オペランド用レジスタとが兼用のため，バスラインが一般のアーキテクチャと異なります。このへんも自作RISCということで見過ごすことにします。

4) ALU

演算器は2つの入力を内部で演算し，その演算結果を出力するものです。これを基本ゲート回路の組み合わせで組むとするととても複雑になり，部品数も配線数も膨大になってしまいますが，これもひとつのパッケージに収められたものが用意されています。ALUはCPUのメインとなる部分ですが，ここをひとつのパッケージで組んでしまうと，わざわざTTLで組む意味も半減してしまいます。とはいえ，図6の回路を全部手配線する元気はとてもなく，ここはやはり楽するしかないと思います。

手頃なのは4ビットALUのLS382で，加減算と論理演算を1個でサポートしています。モード選択端子があり，そこで演算内容を選択します。もちろん桁上げ信号も出ているので，これを2個直列にして8ビットに対処します。

図4　LS160

5) データセレクタとバスドライバ

データを処理する個別的な部品は以上のとおりですが，これらの部品間をデータが往来するためにバスラインを管理しなくてはなりません。そのための部品がデータセレクタとバスドライバです。

まずデータセレクタは，複数の入力からひとつのデータを選択する働きがあります。図7にあるとおり，データセレクタには複数の入力端子とひとつの出力端子とのほかにセレクト端子があり，セレクト端子で指定した入力のデータが出力端子に出てきます。

したがって，たとえば演算結果レジスタとメモリ入出力レジスタとを共用したときに，ALUとメモリのどちらからのデータを保持するかの切り替えはデータセレクタで行えるのです。

今回のRISCでも演算結果レジスタの入力切り替えにはデータセレクタを使うことにします。ただし，実際のパッケージには，8ビット入力のものがなく，4ビット2入力か2ビット4入力かしかありません。そこで，2ビット4入力のLS153を4個並列にして使うことにします。

複数の入力からひとつを選択するには，データセレクタのほかに，先ほどから何度も現れてくる3ステート出力を使う手もあります。3ステート出力とは，通常のHとLの2状態のほかにハイインピーダンス状態Zがとれる出力のことです。

ここで，ハイインピーダンス状態というのは，配線はされていても電気的には断線状態，つまりなにもつながっていないのと同じ状態のことをいいます。それには，各パッケージにあるイネーブル端子で，そのON/OFFを指定してやります。

ですから，図8のように複数の出力をそのまま全部つなげてしまった場合，3ステートでなければ各々の入力が衝突してしまってひとつだけ選択できないわけですが，3ステートであれば選択すべき出力以外はZにしてしまえば，それはなにもつながっていないことなので，目的のデータだけ選択できるのです。

この3ステートを有効に利用すると，回路が簡単になります。というのも，3ステートを持つ素子はそれだけでデータセレクタの役割も持っているので，外付けのデータセレクタの分だけ部品数が浮くのです。

今回の回路では登場しませんが，このほかにバスドライバというものもあります。

図5　LS299

入力						出力	動作
Clear	Mode select		CK	Output control		Q	
	S1	S0		$\overline{G1}$	$\overline{G2}$		
H	L	H	⊔	L	L	–	右シフト
H	H	L					左シフト
H	H	H				Z	ロード
H	L	L				–	ホールド
⊔	X	X	X				クリア
–				H	X	Z	–
				X	H		

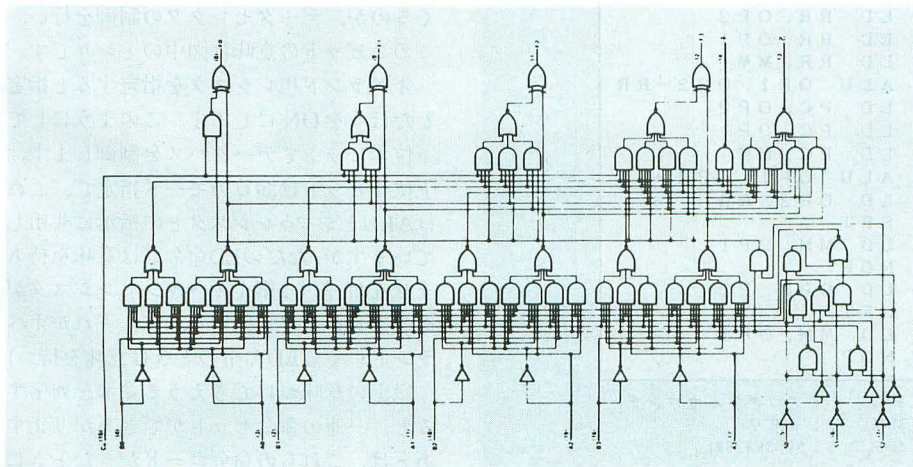
バスのセレクトのほかに，入出力の方向を逆転させて双方向バスとして使えるようにするパッケージもあります。

6) デコーダ

これまではデータが実際に流れる部分の回路について述べてきましたが，忘れてはならないのはそれらのデータバスを制御する部分です。命令コードを読み込んでからはまずその動作を解釈しなければなりませんが，その働きをするのがデコーダです。

デコーダも基本ゲート回路の組み合わせで代表的なものは図9のようなものです。これは入力端子に3ビット2進データ(0～7)を入れるとその番号に対応する出力のみONにします。たとえば，レジスタに各々番号をつけておいて，命令コードのオペランド指定の部分にその番号を書き込んでおくと，デコーダがその番号を解釈してそこで指定されたレジスタのみにONの信号を送るという使い方ができます。今回は2ビットデコーダのLS139を使います。

できあいのパッケージになっているのは，いま述べたような数値データを各番号の出力に振り分けるもの（デマルチプレクサともいう）しかなく，そのほかのデコード部分は自分でゲートを組み合わせて作るしかありません。

組み合わせ論理回路というのはほとんどパズルの世界で，この部分の設計にほとんど頭を使い果たすといっても過言ではありません。これは，命令コードをどう取るかとも関係しているので，わかりやすい命令コード体系をまず設計し，それに合わせてデコーダ回路の仕様（組み合わせ）を決定するのが筋です。今回の設計もひとつの例に過ぎず決して最適化されているとはいえないので，余力のある皆さんはぜひ自分で命令コードとそれを実現するデコーダ回路とを考えてみてください。

7) I/O

このRISCでは，外部のメモリとのやり取りは人間がLEDを見て，トグルスイッチでセットする仕組みですから，そのためのLEDとスイッチを用意しなくてはなりません。LEDやトグルスイッチそのものについては説明は不要でしょう。ところで，LEDはa)次に読み込むべき命令のアドレスを示すプログラムカウンタの表示，b)レジスタからメモリへの転送時にその転送先のアドレスの表示，そしてc)転送するデータの表示の計3カ所に必要です。

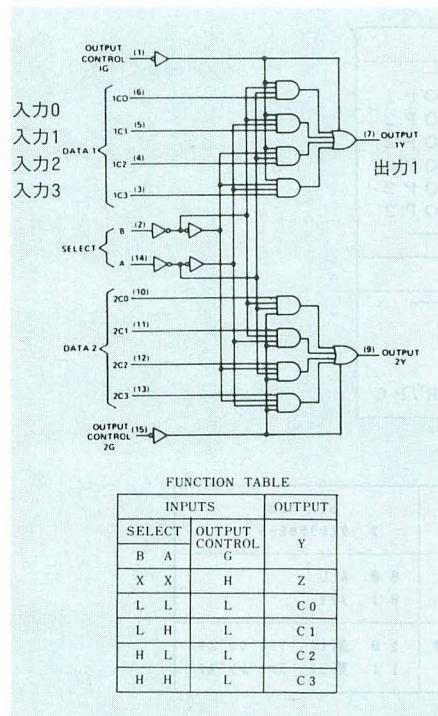
しかし，プログラムカウンタもメモリアドレス指定そのものですから，上のa)とb)は同じところに出てくるはずです。本来は，いったんプログラムカウンタの中身をメモリアドレスレジスタにロードしてから命令コードの読み出しにいくという多ステップ動作をしているのですが，このRISCの場合はプログラムカウンタと演算結果レジスタを並列につないで共通のバスで出力することで，a)とb)とを同じ表示でまかなっています。

また，今回はメモリにデータをセットするための出力レジスタと第1オペランド用レジスタとを兼用していますが，この第1オペランド用レジスタは入出力が同じバスなので，偶然b)とc)も同じところに出てき

図7 LS253

FUNCTION TABLE

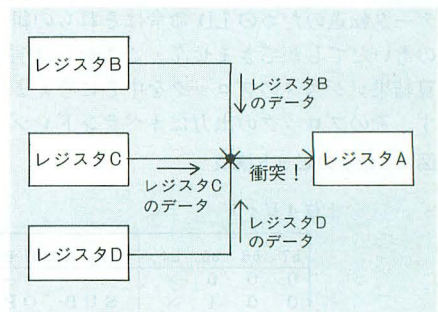
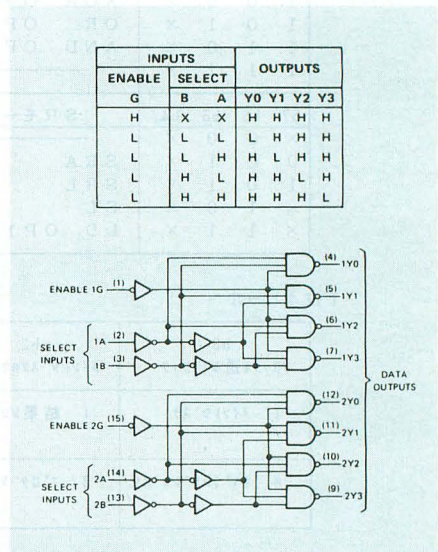
INPUTS			OUTPUT
SELECT		OUTPUT CONTROL	
B	A	G	Y
X	X	H	Z
L	L	L	C0
L	H	L	C1
H	L	L	C2
H	H	L	C3

図8 データバスがぶつかる

図9 LS139

INPUTS			OUTPUTS			
ENABLE	SELECT					
G	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

ENABLE 1G (1), SELECT INPUTS 1A (2), 1B (3), (4) 1Y0, (5) 1Y1, (6) 1Y2, (7) 1Y3, ENABLE 2G (15), SELECT INPUTS 2A (14), 2B (13), (12) 2Y0, (11) 2Y1, (10) 2Y2, (9) 2Y3, DATA OUTPUTS

図6 LS381

SELECTION			ARITHMETIC/LOGIC OPERATION
S2	S1	S0	
L	L	L	CLEAR
L	L	H	B MINUS A
L	H	L	A MINUS B
L	H	H	A PLUS B
H	L	L	A ⊕ B
H	L	H	A + B
H	H	L	AB
H	H	H	PRESET

ます。したがって，LEDは8ビット1組で済んでいます。

自作RISCの回路構成

では，いま説明した部品をどう構成するか，その全体像をつかんでみましょう。もう一度回路図（図2）を図1の命令セットの表と見比べながら見てください。

レジスタは演算結果レジスタとプログラムカウンタとの組とオペランド用レジスタの組との2つのブロックに分かれていて，データ転送のためのLD命令はそれらの組のあいだでしかできません。ここからは演算結果レジスタのブロックを中心に考えます。そのブロックの出力はオペランドレジスタの入力につながっていて，このバスⒶにはプログラムカウンタの中身，アドレスレジスタ（演算結果レジスタと兼用）の中身が出てくるので，ここにLEDをつなぎます。演算結果レジスタのブロックの入力にはオペランドレジスタから，ALUから，そしてメモリからのデータが流れ込んでくるので，ここにデータセレクタを置いて各部からの入力を選択するようにします。

あと，ALUの2つの入力には第1，第2オペランドレジスタがつながっているはずです。このとき，第1オペランドレジスタ（シフトレジスタ，さらにメモリ出力レジスタと兼用）の出力が，入力と同じため，第1オペランドレジスタからALUへのバスⒷは演算結果レジスタから第1オペランドレジスタへのバスと重なっています。

以上のように，データが流れる部分は，三沢氏の解説にあったハードウェア的アーキテクチャの説明図をそのまま引き写しただけであまり考えることなくつなげるのですが，問題はこれらの素子を命令コードに従ってON/OFFさせる制御回路ロジックをどう設計するかです。そこでまず命令コードの構成を考えます。すべての命令はよく考えてみるとあるレジスタにデータをセットするのが基本だということに気づきました。

つまり，LD命令はまさしくデスティネーションのレジスタにデータをセットし，ALU演算命令も演算結果レジスタに結果をセットし，シフト命令にしてもシフトレジスタ内でシフトしたデータをセットし直すということです。ですから，命令コードに従ってどのレジスタにデータセットのクロックを送るか制御すればよいことになります。

あとはオペランドに従ってどこからデータを出力するか選択し，その素子をONにしてやればよいのです。また同時に，データセレクタのセレクト信号も制御する必要もあります。

このような考え方でかなり命令コードが整理されました。図10を見てください。データの流れは結果演算レジスタのブロックに入ってくるかそこから出ていくかのどちらかなので，まずその方向を第4ビットで指示します。これによってまずどちらのブロックにクロックを送るか大まかに分けます。

そして，そのブロックには結果演算レジスタとプログラムカウンタの2つがあるので，そのどちらがオペランドにくるかを第3ビットで指示します。これでどちらのレジスタがONになるか決まります。第2，1ビットはそのブロックの入力がどこからくるのか，データセレクタの制御を行い，この2ビットの意味は図中のとおりです。

オペランド用レジスタを指定すると指定したほうをONにします。このようにして下位4ビットでデータパスを制御します。上位4ビットは演算のモード指定で，これはALUとシフトレジスタとの指示に共用していますが，ただのLD命令では意味を持ちません（ただし，第1オペランドレジスタがシフトレジスタと兼用なので，それがオペランドにくるLD命令のときは意味を持つ）。

以上の意味づけで考えうる命令を列挙すると，一連の命令セットができあがります。あとは，これらの命令コードがきたときに

図10　命令のビット構成

上位4ビット

b7	b6	b5	b4	ALUモード
0	0	0	×	――――
0	0	1	×	SUB　OP2, OP1
0	1	0	×	SUB　OP1, OP2
0	1	1	×	ADD　OP1, OP2
1	0	0	×	XOR　OP1, OP2
1	0	1	×	OR　OP1, OP2
1	1	0	×	AND　OP1, OP2
1	1	1	×	――――

b7	b6	b5	b4	SRモード
×	0	0	×	――――
0	0	1	×	SRA
1	0	1	×	SRL
×	1	0	×	SL
×	1	1	×	LD　OP1, RR/PC

下位4ビット

b3 クロックを送るブロック	b2 メインレジスタのセレクト	b1 b0 データセレクタのモード
1　メインレジスタ	1　結果レジスタ	0 0　ALU 0 1　メモリ
0　オペランドレジスタ	0　プログラムカウンタ	1 0　第1オペランドレジスタ 1 1　第2オペランドレジスタ

b7 b6 b5	b4	b3	b2	b1	b0	
×	×	1	1	1	1	LD　RR, OP2
×	×	1	1	1	0	LD　RR, OP1
×	×	1	1	0	1	LD　RR, MM
ALU	×	1	1	0	0	ALU　OP1, OP2→RR
×	×	1	0	1	1	LD　PC, OP2
×	×	1	0	1	0	LD　PC, OP1
×	×	1	0	0	1	LD　PC, MM
ALU	×	1	0	0	0	ALU　OP1, OP2→RR
×	×	0	1	1	1	LD　OP2, RR
SFT	×	0	1	1	0	SFT
×	×	0	1	0	1	LD　MM, OP1
×	×	0	1	0	0	NOP
×	×	0	0	1	1	LD　OP2, PC
SFT	×	0	0	1	0	SFT
×	×	0	0	0	1	LD　MM, OP1
×	×	0	0	0	0	NOP

RR　：演算結果レジスタ　　PC　：プログラムカウンタ
OPn　：第nオペランドレジスタ　　MM　：メモリ
ALU　：ADD, SUB, AND, OR, XOR　　SFT　：SL, SRA, SRL

各ビットの組み合わせて目的の動作を実現する制御信号を作ればよいのです。

デコーダ回路の実際

デコーダ部の回路図を見ると，LS139のほかにANDとORとNOT（とXOR）とでできています。LS139の働きはすでに述べましたが，残りの部分に説明を少し加えます。と，その前に正論理と負論理について述べる必要があります。論理演算は1（正）と0（負）との2値で行われ，たとえばAND演算を表にすると図11a)のようになります。このときTTLのHレベルを1，Lレベルを0に対応させるとちょうどLS08の機能と同じになりますb)。それでLS08のことをAND回路というのです。しかしここで，Lレベルを1，Hレベルを0と対応させてみます。すると，同じ演算でありながら，今度は別の機能になってしまいますc)。本来OR回路であるはずのものがAND演算を実行してしまうのです。

この理由は1/0とH/Lレベルの対応を逆転させたところにあります。この対応をそれぞれ正論理，負論理というのです。いまの例でいうとOR回路LS32は負論理ではANDになるということです。記号の上では図12のようになります。基本的にブーメランのような記号がOR，蒲鉾のような記号がANDを意味し，それぞれの入出力において負論理のときに丸をつけるのです。同じLS32でも正論理のときはORであり，負論理のときはあくまでもANDなのです。

デコーダ部の回路図の中でもNAND回路が正論理，負論理両方の意味で使われているのに気づくことでしょう。正論理のときは「入力が両方Hのときだけ出力はL」と考え，負論理のときは「入力がどちらか一方Lならば出力はH」と考えるのです。そこで，このような論理回路を追うときの着眼点は，ANDなら「条件が揃わないと次に続けない場合」，ORなら「条件がどちらか満たせばそれで次に続ける場合」と覚えておくのがよいでしょう。HとLの組み合わせで機械的にとらえるのは得策ではありません。

以上の論理の違いを頭に入れたうえで，デコーダ部について見てみましょう。それには具体的にひとつの命令コードについてシミュレートするのがわかりやすいと思います。では，LD RR，OP1（$XXXX1110：ただし命令コードは正論理にとっているので，これはXXXXHHHLと同じ）の一部分を追ってみます（図13）。

図11　ANDとOR

a）ANDの論理

	0	1
0	0	0
1	0	1

b）LS08の論理

	L	H
L	L	L
H	L	H

c）LS32の論理

	L	H
L	L	H
H	H	H

d）ORの論理

	0	1
0	0	1
1	1	1

図12　正論理と負論理

図13　デコードの様子

これで完成

第3ビットはHなので1のANDにH，3のNOTを通過したあとは2のANDにLが入ります。1，2のANDにはクロックが入ってきますが，2のほうは出力がHのままになってしまい，クロックが遮断されてしまいます。1のANDに入ったほうは正負逆になりますが，クロックが抜け，さらに4と5のANDに入ります。第2ビットもHなので6のNOTを通過したあとの4のANDにL，5のANDにはそのままHが入ります。4，5は負論理のANDなので，Hの入る5のほうは出力がLのままになってしまい，クロックが遮断されてしまいます。6のANDに入ったほうは正負が戻ってクロックが抜けます。

こうして演算結果レジスタにだけ選択的にクロックが入り，目的の動作が実現するのです。第0，第1ビットの行方も論理の違いに注意しながら追ってみてください。

いまシミュレートしたような手順で順番に注意深くデコーダ回路を設計していくのです。これは職人芸的なところがあり，ある程度の経験がないと手がつかないかもしれません。一般のパソコンユーザーの皆さんには，論理回路の組み合わせでデコーダ部を構成するときの様子がなんとなく伝われば十分だと思います。

しかし，もしハードウェアを自分のものにしようという意気込みがある人にとっては，ほかの人の設計した回路図をよく読んでその動作を追うのが勉強になります。いろいろな回路に数多く接して，カンを養っていくのが近道でしょう。

CPUを製作する

やっとのことで製作するCPUの仕様が決まりました。最初に構想を練ったときは実体配線図を載せて皆さんにも作ってもらえるような回路をと思っていましたが，欲張りすぎて，ちょっと試してみるというのに

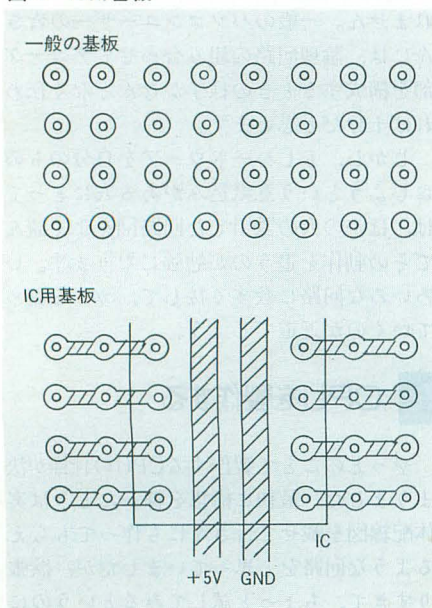
はほど遠い回路になってしまいました。そこで，皆さんにはまた別の機会にもっと配線数の少ない回路を紹介することにして，今回はこういうデジタル回路を組むときのコツみたいなものをお話しして終わりたいと思います。

電子工作をするには，まず工具を揃えなければなりません。工具をケチると工作に苦労するうえ，できばえも悪いものにしかなりません。最低限必要なのは，ハンダゴテ，ニッパ，ラジオペンチ，カッター，ドライバ，ワイヤストリッパ，ピンセットの7点です。これらの七つ道具を選ぶうえでの注意点を列挙しますと，

1) ハンダゴテは，IC工作用のコテ先の細い20W程度のものがベストです。ハンダもそれに合わせて細いものを選びます。また，スポンジのついたコテ台もセットで揃えましょう。このスポンジでコテの先を綺麗にしながらハンダづけしていくのですが，コテ先が汚いと絶対にハンダづけは失敗します。

2) ニッパとラジオペンチとワイヤストリッパとを別々に揃えることをすすめます。一見ペンチだけで代用できそうですが，それぞれの役割があり，作業の能率が全然違います。あまり安物にしないで，最初からそれぞれ2,000円程度のものを買っておくのがよいでしょう。

3) IC工作は部品が小さく作業もこまかいので，ピンセットも必需品です。これは専用のものでなく，文房具屋で手に入るもので構いません。ただし，先の細いものを。

4) ドライバはプラスとマイナスの太さの違うものをそれぞれ3本ずつぐらいほしいと思います。できたら，手でにぎるタイプのもののほかに精密ドライバ（腕時計の修理などに使うもの）と呼ばれるセットも買い揃えておくと便利です。

工具が揃ったら，次は部品集めです。秋葉原（または類似品）が近い人でなければ，通信販売に頼るしかありません。しかし，作っていく途中で，あれが足りない，これが足りないということがよくあります(特に自分で設計しながら作っていく場合)。ですから，部品表を載せている製作記事のものを作るようにしたほうが賢明です。部品表をそのままコピーして販売店に送れば，まず間違いがないでしょう。

実際の部品について気づいた点を述べておきます。まず製作のうえでポイントとなるのは，基板選びです。ICの足が密集しているために，ハンダゴテの先が当てにくく苦労します。そこで，基板のプリントパターンに余裕のあるものがベストです(図14)。これならひとつの足に複数の配線があるときに便利で，特に今回のCPUのようにバスラインがタコ足に交差している回路の場合このタイプの基板以外考えられません。また，IC用の基板には＋5VとGNDのラインが付いているものが多く，各ICの電源ラインの配線に便利なようになっています。

基板が決まったら，次に部品のレイアウトを考えなければなりません。これには，ICソケットを実際に基板上に並べてみて，配線のしやすさをチェックします。うまくレイアウトされた基板は，部品の並び方を見ただけで信号の流れがつかめるものです。今回のCPUについても，アーキテクチャの面から3つのブロックに分かれているので，ブロックごとに部品を配置していきます。

レイアウトするときも，スペースをゆったりとることが大切です。というのも，あとから回路を変更したり，付け足したりすることもよくあるからです。部品のレイアウトが決まれば，あとはただひたすら配線するのみです。このときも信号の流れを追いながら，ブロックごとにまとまりをもって配線していくのが効率的です。まず全体の頭脳となるデコーダ部の配線，スイッチやLEDの周辺部の配線のあと，単調なバスラインの配線へと移ります。

一度に配線を終わらせようとしないで，小ブロックごとに動作をチェックしながら組み立てていくほうがよいかもしれません。回路図ではバスラインを8本ひとまとめに1本線で書いてあるので，間違えないように。しかもICのピン番号を明記していないので，規格表で確認しながら作業します。IC工作ではピン数が多いのでいちいちピン番号を書かないこともよくあることです。

図14 IC用基板

次回こそは製作実習を

今回は三沢氏のCPUのアーキテクチャの解説に触発されて，身近なTTL ICでCPUの内部を再現してみようと試みました。アーキテクチャの構築に始まって，全体的な回路の立案，部品の選び方やその組み合わせ方，それに，機械語の命令をハードウェアで実現していくデコーダの仕組みとその設計法にいたるまで順を追って説明したつもりです。この記事をとおして，ハードウェア設計のアプローチの仕方がわかってもらえたのではないでしょうか。でもこれだけではかえってハードは難しいと思われてしまうかもしれません。しかし，どんなに複雑そうに見えるハードウェア設計も簡単な回路の組み合わせにすぎないのです。

皆さんも一度ハード工作に触れてみたら，意外と簡単だったなあときっと驚くに違いないと思います。そこで次のハード特集のときには，皆さんも一緒に工作できる回路を用意して実際に動かしながら読んでもらえる企画をお届けします。ICが全部で4個ぐらいの構成で，完全部品表と実体配線図が付いて誰にも作れ，しかも実用的でユニークな回路の製作入門記事にする予定です。では，そのときまたお会いしましょう。

◆参考文献
'89TTL規格表（CQ出版社）

表1 部品表(値段は単価，秋葉原調べ)

	品名	個数	単価
TTL	LS374	2	115
	ALS561	2	250
	LS299	1	395
	LS382	2	300
	LS253	4	70
	LS139	1	60
	LS00	2	30
	LS04	1	30
	LS02	1	30
	LS08	1	30
	LS86	1	40
スズメッキ線/ビニール導線		適量	

品名		個数	単価
トグルスイッチ		16	80
LED	赤	8	30
	緑	3	30
押しボタンスイッチ		1	100
ICソケット	14pin	6	50
	16pin	5	35
	20pin	7	45
IC用基板		1	900
抵抗	10kΩ～	20	10
	220Ω	11	10
コンデンサ	0.01μF	若干	30
計			約6300円

micro computer 入門

業界初！

EDSACプログラミング入門

世界初の“プログラムできる”コンピュータ，EDSAC。現在では当たり前の考え方となりましたが，温故知新の意味を含めてそのアーキテクチャについて解説します。きっと最新のコンピュータとは違った新しい何かが見つかることでしょう。

Miyajima Yasushi
宮島　靖

計算機の祖先

第２次世界大戦のさなか弾道計算を行うために、ペンシルベニア大学でエッカート教授とモークレー教授により，真空管１万８千本を使用したコンピュータ「ENIAC」が開発されました。１万８千本の真空管に同時に通電するため，付近の民家に電気が行かなくなったという話もあります。ENIACは現在のコンピュータとはプログラミングスタイルがまったく異なり，操作板の配線をつなぎ変えることによって，いろいろな動作をするようになっていました。ハードウェアに密着したプログラミングです。

もっと，プログラム効率のよい方法がないかということで研究を進めていたのが，かのJ.von Neumann（フォン・ノイマン）です。読者の人も聞いたことがある名前でしょう。彼はプログラムを記憶装置（平たくいえばメモリ）の中に入れてしまい，これをひとつずつ順番に取り出していき実行するStored Program方式を提案しました。実は現在のポケコン，パソコン（当然X68000も），大型機も，基本的にこの方式を採用しており，このような方式のことを彼の名前を取って，「ノイマン型コンピュータ」といいます。

ノイマン型の重要なところはメモリ上に入っているビット列は命令であるのか，単なるデータであるのかが見分けがつかず，Scc(Sequence Control Counterの略，現在はPC:Program Counterというほうが一般的）の指し示す番地が命令になるという点です。

そして世界で最初のノイマン型のコンピュータが1949年にイギリスのケンブリッジ大学でM.V.Milks教授によって生まれました。主記憶には超音波水銀遅延線メモリを用い，紙テープによってプログラムをロードするという方式のコンピュータです。この機械はElectronic Delay Strage Automatic Calculatorの頭文字を取って「EDSAC（エドサックと発音すればよい)」と呼ばれることになりました。まさしく，EDSACこそが現在のコンピュータの祖先であり，プログラム内蔵方式もこの機械で初めて実現されたわけです。

EDSACのアーキテクチャ

というわけで，EDSACの生まれた背景はわかっていただけたでしょう。では次に，EDSACのアーキテクチャ寄りの話をしたいと思います。

a）レジスタ構成

図１にEDSACのレジスタ構成を示します。EDSACには，Acc（アキュムレータ)，Mレジスタ，Bレジスタの３つしかレジスタがありません。しかも，Accは固定小数点の累算器，Mレジスタは掛け算用の固定小数点レジスタ，Bレジスタは番地インデックス用の整数レジスタというようにしか使用できません。

Z80や68000のアセンブラに慣れている人には非常に貧弱に見えるでしょう。確かに，EDSACでのプログラミングはいかにして計算途中のデータをうまく保存しておくかがポイントになり，68000のようにレジスタを巧みに利用したプログラミング環境とはまったく様相を異にします。

まずはAccですが，16ビット，32ビットなどとケチなことはいいません。最新64ビットCPUを凌ぐ70ビットもあります。が，メモリに転送するときには先頭の17ビットか，長語（後述）を使用しても先頭から35ビットしかメモリに転送することができません。

ではなぜ，70ビットもあるかというと，Mレジスタとメモリの内容の乗算を長語で行った場合，結果は70ビットになるのですが，上から35ビットで切ってしまうと，誤差が出ることになるからです。乗算を行った結果を足し込んでいくような場合や，乗算結果を左にシフトする場合など，下位35ビットがないと誤差が出てくるわけです。

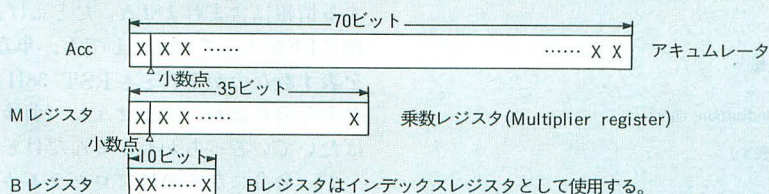
次にMレジスタですが，このレジスタは基本的に乗算命令以外には使用されません。EDSACでは即値との乗算ができないので，このレジスタを用いるわけです。

Bレジスタは，ほかの２つのレジスタとは異なり，番地のインデックス用なので整数しか入りません。後述しますが，EDSACの番地は1023番地までなので，Bレジスタも10ビットの幅が用意されています。

b）EDSACの扱える数値

EDSACはメモリの１ワードを17ビットとして扱い，レジスタはAccが70ビット，Mレジスタが35ビット，Bレジスタが10ビットとなっています。

EDSACでは基本的に小数しか扱うことができません。メモリの内容にしろ，レジスタの内容（ただし，Bレジスタは除く）にしろ，$-1 \leqq x < 1$までの範囲しか代入できないのです。最上位ビットと２番目のビットのあいだに小数点があると考えます。また，負の数は２の補数表現で表されるので，最上位ビットは符号ビットとなります（図２）。

図１　EDSACのレジスタ

c) EDSACのメモリ空間

EDSACにはメモリが0～1023番地までの1024語分が実装されています。前述したように，1語は17ビットなのでほぼ2Kバイトのメモリといえるでしょう。

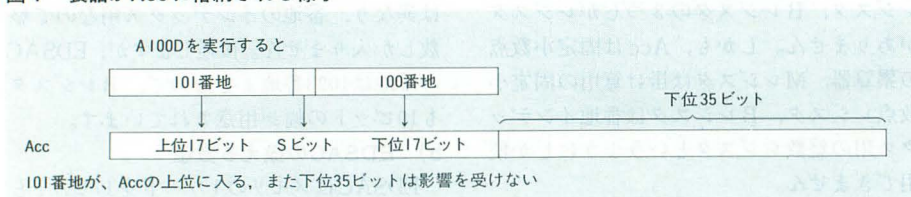
先ほどちょっと書きましたが，EDSACには長語という概念があります。ちょうど68000でいうロングワードのようなものです。68000では1ワード16ビットなので，長語は2倍の32ビットとなりますが，EDSACではなぜか35ビットになります。17×2＝34なのに，なぜ1ビット増えたのでしょうか？

これは，実はEDSACにはSandwich Digitと呼ばれるビットがあって，奇数番地と偶数番地の間に1ビット分はさまっているビットがあるのです。このビットは本来はメモリ読み出しのタイミングに使用されているビットなのですが，このビットを飛ばして2番地分読むよりも，一緒に読んでしまうほうがラクなのよということで，読まれることにあいなったものなのです。

図3にメモリの番地割り付けを図示しましたが，注意すべき点は，奇数番地が先頭にきているというところです。1，0，3，2，5，4……というぐあいに並んでいるため，Accから2n番地に書き出したりする場合には，Accの上位17ビットが2n＋1番地に，下位17ビットが2n番地に入り，18ビット目がサンドイッチビットに入ります。この様子を図4に示しました。

もしかして，インテルの上位下位逆転のアーキテクチャは，EDSACをお手本にしたのだろうか？（違うって）

図2　数値表現

図3　EDSACのメモリ

図6　命令の例

図4　長語がAccに格納される様子

図5　命令表現

命令はSingle-address型であり，短語の1語で下記のように表す。

	5ビット	1ビット	10ビット	1ビット
	0 1 2 3 4	5	6 7 8 9 10 11 12 13 14 15	16
命令の表現	X X X X X	X	X X X X X X X X X X	X
	命令部 (Function digits)	番地修飾部 (B-digit)	番地部 (Address digits)	特殊指示部 (Special indication digit)

命令部：命令の種類を表す。命令の一覧を43ページに示す（1文字で表す）。
番地部：引用する番地を示す。
番地修飾部：1のときは番地部にBレジスタの内容を加える。0のときは何もしない（1のときSで表す）。
特殊指示部：1のときは長語の指定，または命令の修飾に使う。0のときは短語の指定

d) EDSACの命令について

EDSACの命令は，1語17ビットの中に命令と，それに必要な番地データまたは数値データ，そして修飾用ビットが含まれています（図5）。

命令部は先頭の5ビットであり，命令のビット列は表1に示すような内部表現になります。

5ビット目の番地修飾部が1のときには，Bレジスタの値を番地部に足し込んだ値を実効アドレスとするというスイッチです。1のときには「S」と表記します。

6ビット目からの10ビットが番地部と呼ばれ，アドレスが格納されている場所です。10ビットの幅があるので1023番地までを指定することができます。

最後の1ビットが特殊指示部と呼ばれ，1のときには長語指定か，命令の修飾に使用します。長語を意味するときには，「D」と表記し，命令の修飾に使用するのならば「D」または「S」を使用します。また，このビットが0のときには，短語指定となり，「F」と表記します（図6参照）。

プログラミング序論

命令だけ見てもピンとこないでしょうから，具体的な例をあげて「こういうときはこうする」といった定石を説明しましょう。

その前に，軽く命令とデータの扱いについて説明しておきます。

Z80でも68000でも，メモリに入っているビット列には，命令か単にデータかを区別する情報は含まれません。たとえば，100番地にFFが入っていたとしても，単なる255を表す数なのかそれともRST 38Hという命令なのかはわかりませんね（前後を見ればたいていどっちかわかるんだけどね）。

で，命令になるのはプログラムカウンタがそこの番地にきたときなわけです。EDSACでも同様で，

01000000000000000

というビット列があったとき，0.5にもなるし，ＩFという命令にもなります。EDSACでは，Accに命令を作ってそれをメモリに転送して実行するような自己命令書き換えを当然のように使用します。現在のMPUでは，アドレス空間も広く，命令も豊富なので自己書き換えなぞはしないほうが望ましく（これをやるとプログラムがわかりにくくなってしまう），命令数の少ないEDSACならではの手法なのです。

●Accのクリアの方法

EDSACにはAccをクリアするという命令はありません。ではどうするのかというと，T命令を使います。この命令は本来はAccをどこかの番地に転送する命令なのですが，同時にAccもクリアします。どこか使っていないメモリにAccを転送してやればAccをクリアできます。

●Accに値を代入する

Accに値をロードする命令は存在しないので，いったんAccをクリアしてから，A命令を使用してAccに足し込みます。

つまり，

```
100 T 110F
101 A 111F
     ⋮
110 P F  *PFは小数に直すと0.0
111 I F  *IFは小数に直すと0.5
```

としてやれば，101番地でAccには，0.5が代入されます。面倒ですが，Accにデータを代入したいときには，いったんメモリを介して行うしかありません。

●ループの作り方

ループにはふつうBレジスタを使用します。たとえば，

```
100 B 50F
101 BS 1S
     ⋮
  n J 101F
```

とすると，101番地からn番地の間を50回ループさせることが可能となります。

●自己書き換えの例

問．100番地に$n \cdot 2^{-15}$が入っているとき，n番地の内容を110番地に格納せよ。

$n \cdot 2^{-15}$というと複雑そうに聞こえますが，要は番地部にnが入っているということです。Accが最初0だとして，

```
200 A 206F
201 A 100F
202 T 203F
203 P    F
204 T 110F
205 Z    F
206 A    F
```

とすればOKです。202番地で203番地の内容を書き換えて実行しています。この様子を図7に示します。EDSACではこのような手法をマスターすることが必要条件ともいえますので，しっかり理解してください。

●サブルーチンの作り方

本来なら，サブルーチンはスタックの概念が必要なのですが，EDSACにはスタッ

EDSACの命令セット

■転送命令

Φn(10100*n*)，Un(00111*n*)，Tn(00101*n*)

Tn，Unはどちらも，Accの内容をn番地に転送する命令ですが，Tnは転送後にAccの値をクリアします。ΦnはTnと同様な動作をしたあと，Accがオーバーフローしていたときに，ベルを鳴らして停止するという命令です。なお，「Φ」は入力できないのでシミュレータ上では「$」で代用します。

Hn(10101*n*)

Hn命令はn番地の内容をMレジスタに転送する命令です。これらの命令はBレジスタによる番地修飾部および，長語の指定も可能です。すなわち，「TS100D」とすると，Bレジスタの値に100を加えた番地へ長語でAccを転送する命令ということになり，101番地にはAccの上位17ビットが，100番地にはAccの下位17ビットが，そして，18ビット目がサンドイッチビットに転送されることになります。くれぐれも，長語指定のときには，上位下位が逆になることを注意しておいてください。また，長語指定のときには必ず実効アドレスが偶数番地になるようにしなければなりません。

BnF(111010n0)，BnS(111010−n0)

どちらも，Bレジスタに直接nを転送する命令です。ただし，BnSの方はBレジスタに−nを代入します。しかし，実際にはBnSという命令は内部では，B−nFとして処理されています。というのも，nは10ビットであり，負の数は補数で表現するため，−1＝1023であるわけで，要は，nの先頭ビットが立ってさえいれば負の数ということになるのです。つまり，「B2S」は「B1022F」と同じ結果になります。

KmF(011100m0)

この命令は特殊な命令です。m番地に「B(Bレジスタの値)F」という命令を格納するというちょっと複雑な命令です。つまり，Bレジスタの値が100のとき，K200Fという命令を実行すると，200番地に「B100F」という命令が格納されます。一見なんの役に立つのかわからない命令ですが，「サブルーチン」のところで説明します。

■演算命令

An(11100*n*)，Sn(01100n*)

Anは加算，Snは減算命令です。Accとn番地の内容を計算します。

Vn(11111*n*)，Nn(10110*n*)

Vnはn番地の内容とMレジスタの内容を乗算して，Accに加え，NnはAccから引くという命令です。注意すべき点は，Acc に乗算の結果が転送されるのではなく，結果が加算されたり減算されたりするということです。これらの命令はBレジスタ，長語ともに使用可能です。

YmF(00110*m0)

Accに2^{-35}を加えます。つまり，Accの内容を34ビットに丸めるのに用います。なお，mの値は意味を持ちません。

Cn(11110*n*)

n番地の内容とMレジスタの論理積をAccに加えます。

Mn(10111*n*)

Accの先頭から6ビットをクリアして，n番地の内容を加えます。

BSnF(111011n0)，BSnS(111011−n0)

BSnFは，Bレジスタにnという数値を加え，BSnSはBレジスタからnという数値を引きます。この命令も内部では，「BSnS」=「BS−nF」として処理されています。

■シフト命令

Accのビットを左右にシフトする命令です。この命令を使用すると，普段見ることのできない，Accの下位35ビットを見ることができます。

R　D(0010000000000001)，
L　D(1100101111111111)

RとDの間に空白があるのは，番地部が0だからです。R0D，L0Dでも同じですが，nが0の場合は省略してしまうようです。R　Dは右に，L　Dは左へ1ビットAccをシフトします。

R　F(0010000000000000)，
L　F(1100100000000000)

R　Fは右へ15桁，L　Fは左へ13桁Accをシフトします。

$R2^{\rho-2}F$($001000\ 2^{\rho-2}\ 0$)，$L2^{\rho-2}F$($110010\ 2^{\rho-2}\ 0$)

それぞれ右へρ桁，左へρ桁Accをシフトします（$2 \leqq \rho \leqq 11$）。つまり，L8Fで，左へ5桁シフトすることになります。

■分岐命令

分岐命令には，無条件分岐命令と条件分岐命令があります。現代のMPU（CPU）は，フラグを持っており，その値によって様々な分岐ができるようになっていますが，EDSACにはAccやBレジスタが0かどうかを調べて分岐する命令しかありません。

FmF(100010m0)

無条件にm番地へジャンプします。

FmD(100010m1)

Acc≠0のとき，m番地にジャンプします。

FSmF(100011m0)

（Bレジスタ＋m）番地にジャンプします。後述の「サブルーチン」のところで説明します。

EmF(000110m0)，EmD(000110m1)

どちらの命令もAcc≧0のときにm番地にジャンプしますが，EmDのときにはAccをクリアします。

GmF(110110m0)，GmD(110110m1)

どちらの命令もAcc＜0のときにm番地にジャンプしますが，GmDのときにはAccをクリアします。

YmD(001100m1)

以前にYmDが実行されて以来，Accがオーバーフローしていたならば，Accをクリアしてm番地にジャンプします。滅多に使うことはないでしょう。

JmF(010100m0)

Bレジスタの値≠0ならば，m番地にジャンプします。BS1S命令と一緒に用いてZ80のDJNZ命令のような主にループを構成する場合に使用します。

■入出力命令

EDSACのI/Oは紙テープだけです。データやプログラムは5ビット幅の紙テープから読まれます。この5ビットの穴はEDSACの内部表現コード（表1参照）に対応しており，32種類の英字と記号または，単純に0～9の数字として読み込めます。

In(010000m0)

テープ1列を整数xとして読み込み，$x \cdot 2^{-16}$をn番地に格納します。つまり「11100」という5ビットのビット列（これはEDSACのコードで「A」を表します）を読み込むと，n番地には，

0000000000011100

というビット列が格納されます。また，長語で読むことも可能です。

On(010010m0)

n番地の先頭5ビットを紙テープに出力します。

■その他の命令

ZmF(011010m0)

機械を停止します。プログラムの終了に書いておくのがふつうです。mの値は意味を持ちません。

ZmD(011010m1)

EDSACにはZDスイッチと呼ばれるハードウェアスイッチがあって，このスイッチがONのときに，ZmF命令と同様の働きをします。OFFのときは次の命令を実行します。

・括弧内はビットパターン。n，mは10ビットの2進数

クというものが存在しないので，Bレジスタをうまく利用したclosed Bと呼ばれるサブルーチンの作り方を紹介します。

詳しい説明は図8に示しましたので，ここでは考え方について簡単に説明します。

サブルーチンからメインルーチンに戻る場合，戻り番地というのは呼び出された番地の次の番地であるので，戻るべき番地をBレジスタに入れてやって，サブルーチンを呼び出せばうまくいきそうです。そして，呼ばれたサブルーチン内で，FS2Fとしてやれば，B＋2番地にジャンプします。つまりこれがリターン命令の役目を果たします。

しかし，この方法だとサブルーチン内でBレジスタが使用できなくなってしまいます。そこで用意されたのがKmFという命令です。この命令を実行するとm番地に，現在のBレジスタの内容を，Bレジスタに代入する命令が格納されます。たとえば，Bレジスタの値が123のときに，KmFという命令を実行してやると，m番地には，「B123F」という命令が格納されます。KmFをサブルーチンの先頭に置いて，m番地をサブルーチンのいちばん最後の番地にしてやれば，サブルーチンに飛んできてすぐに，Bレジスタを間接的にm番地に待避したことになるので，なかでBレジスタを使用しても問題がなくなるわけです。

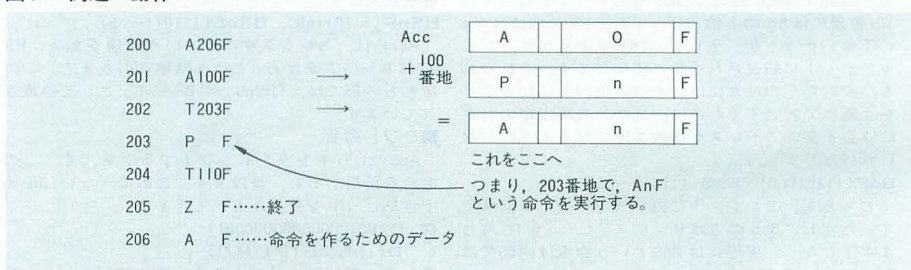
表1　EDSACの入出力符号

入力符号		
内部表現	キーボード上の記号	
	F.S.	L.S.
0 0 0 0 0	0	P
0 0 0 0 1	1	Q
0 0 0 1 0	2	W
0 0 0 1 1	3	E
0 0 1 0 0	4	R
0 0 1 0 1	5	T
0 0 1 1 0	6	Y
0 0 1 1 1	7	U
0 1 0 0 0	8	I
0 1 0 0 1	9	O
0 1 0 1 0		J
0 1 0 1 1		π (%)
0 1 1 0 0		S
0 1 1 0 1		Z
0 1 1 1 0		K
0 1 1 1 1		Erase (&)
1 0 0 0 0		,(Blank Tape)
1 0 0 0 1		F
1 0 0 1 0		θ (♯)
1 0 0 1 1		D
1 0 1 0 0		Φ ($)
1 0 1 0 1	＋	H
1 0 1 1 0	−	N
1 0 1 1 1		M
1 1 0 0 0		Δ (@)
1 1 0 0 1		L
1 1 0 1 0		X
1 1 0 1 1		G
1 1 1 0 0		A
1 1 1 0 1		B
1 1 1 1 0		C
1 1 1 1 1		V

(　)はASCII文字を示す。　F.S.：Figure Shift　L.S.：Letter Shift

世界最短アセンブラ

さて，EDSACのプログラムを入力するにはどうしたらよいでしょうか。17ビットのスイッチを使用して1番地ずつメモリに値を書き込んでいったのでは，日が暮れてしまいます。そこで，EDSACには簡単なアセンブラが用意されています。このアセンブラは紙テープから5ビットのEDSACコードを命令，番地，制御コードの順に読んでいき，17ビットの命令に変換して格納してくれ，さらに相対番地までサポートしているというくせに，ナントまあ，全体で54ワードの長さという驚異的なアセンブラです。最近あまり見かけませんが，一時期，雑誌に256バイト以内でなにかプログラムを作ったり，BASICで1行のプログラムを作ったりという，いかに短いプログラムで高機能なものを作るかということに燃えていた人人がいましたが，機能・芸術性でこのアセンブラにかなうものはないでしょう。

このアセンブラのことをInitial Input Routine（初期入力ルーチン）と呼んでおり，EDSACでは0番地から53番地にロードされます。そして，0番地から実行を開始すると，紙テープから順次5ビットずつキャラクタコードを読み取っていき，メモリに格納して指定番地に制御を移します。

初期入力ルーチンに対する制御指令を次に説明します。

PZ

プログラムの先頭だということを示します。

TmK

以下の命令をm番地から格納します。T100Kと書くと，その後ろに続くテープを100番地から格納し始めます。Z80のアセンブラのORG疑似命令とでも思ってください。

GmK

相対番地の基点を示します。TmKのあとに続けて書くときはGKとするだけでも構いません。

メモリに読み込まれた状態のプログラムは，すべて絶対番地形式であり，ほかの番地へ移動すると実行できなくなります。そこで，紙テープに書くときには相対番地で書いて，初期入力ルーチンに番地を割り付けさせながら格納する方法が開発されました。この方法により，GmKで指定した番地からのオフセットでニーモニックを記述することができ，TmK，GmKのmの値を変更するだけで，任意の番地での実行が可能となります。なお，ニーモニックでの記述にはF, Dの代わりにθ, πθを用います。ただし，シミュレータ上では♯，%♯を用います。

たとえば以下のような，

PZ（プログラムの先頭）

T100K（100番地から格納）

図7　例題の動作

図8　closed Bサブルーチンの概念

つまり，Bレジスタに（呼び出しを行う番地−1）を入れ，コールする。そして，サブルーチンの先頭で，Bレジスタに，呼び出されたときのBレジスタの値を入れる命令をサブルーチンの末尾に書き込む。

サブルーチンの最後で，呼ばれたときのBレジスタが，復活するので，FS2F命令によって，(呼び出し番地＋1)に戻ることができるのである。

GK(100番地が基底)
A10#
T20%#
A0F
ZF
E100KPF(100番地へ)

を読み込ませると,

100 A 110F
101 T 120D
102 A 0F
103 Z F

というぐあいに読み込まれます。

EmKPF

Accをクリアしてm番地に制御を移します。

まだこのほかにもいくつかあるのですが,これだけ知っていれば十分でしょう。

えどさっ君利用の手引き

さて,紙面だけではよくわからないと思われるので,EDSACのシミュレータを作成しました。かなりの大きさになってしまったので,掲載はできませんでしたが梁山泊ネットとPEKINネットに「えどさっ君」をPDSとしてアップしておきますので,会員の方はそこからダウンして使ってください。

EDSACのソースプログラムは,本体付属のEDや市販のエディタなどの標準テキストファイルを作成できるもので作成します。

そのとき,次の点に注意してください。

・行番号,番地番号は付けない。
・余分な空白は入れない。
・すべて大文字で記述する。

正しい例	悪い例
PZ	98 PZ
T100K	99 T100K
AF	100 A F
T105F	101 T 105F
ZF	102 Z F
E100KPF	103 E100KPF

えどさっ君では2種類の初期入力ルーチンをサポートしています。高速初期入力ルーチン(HIPL),標準初期入力ルーチン(IPL)の2つです。

HIPL:独自の手法で記述された初期入力ルーチンであり,1命令ずつ直接読み込んでメモリに格納するため高速。ただし,現在サポートしている制御命令はPZ,TmK,GmK,EmKPFと相対番地指定の#だけです。これ以外の制御命令を使用している場合はIPLを使用して読み込んでください。

IPL:IPLは,本物(?)の初期入力ルーチンがそのまま動いているので,すべての制御命令が使用できますが,その反面速度が遅くなります。

コマンド一覧

“EDSAC>”と表示されているときに,なにも入力せずにリターンキーだけを押すとコマンドの一覧が表示されます。また,各コマンド名だけを入力してリターンキーを押すと,そのコマンドについての詳しい説明が出てきます。なお,コマンドは小文字でも,大文字でもどちらでも構いません。スイッチの順番も任意です。ただしスイッチとスイッチまたは,スイッチとパラメータのあいだはスペースを1文字分入れてください。

BREAK アドレス

機能 ブレイクポイントを設定する。JMP命令,IPL命令を実行中にPC(プログラムカウンタ)が設定値にくると,実行を中断してコマンドラインに戻る。

例 EDSAC>BREAK 100

とすると,100番地で実行が中断するように設定される。

CLR

機能 全レジスタをクリアする

例 EDSAC>CLR

Acc=0,Mreg=0,Breg=0になる。

DISASM アドレス1[アドレス2]
[スイッチ1]……[スイッチn]

スイッチ
/C……画面への出力はしない
/F ファイル名
……ファイルへの出力
/P……プリンタへの出力
/W……2段組み出力

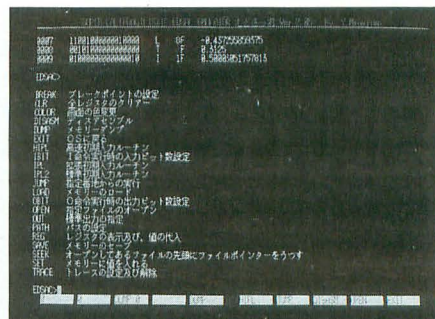
機能 アドレス1からアドレス2までの内容を出力する。

例 EDSAC> DISASM 0 9 /F BAKA /W /P

0000	T	F	0005	R	4F
0001	E	20F	0006	V	F
0002	P	1F	0007	L	8F
0003	U	2F	0008	T	F
0004	A	39F	0009	I	1F

と画面に表示され,同時にプリンタとファイルにも出力される

DUMP アドレス1[アドレス2]
[スイッチ1]……[スイッチn]

スイッチ
/B……2進の表示をしない
/C……画面への出力をしない
/F ファイル名
……ファイルへの出力
/H……16進表示を追加

/O……8進表示を追加
/P……プリンタへの出力

機能 メモリのダンプを行う

例 EDSAC>DUMP 100 200 /H /O

EXIT

機能 親プロセスへ戻る。

HIPL ファイル名 [スイッチ]

スイッチ
/I……読み込んだ命令の表示

機能 高速初期入力ルーチンの実行。あらかじめ作成してあったソースファイルを読み込む。読み込める制御命令はPZ,TmK,GmK,EmKPF,……#のみ。EmKPFを読み込むとコマンドラインへ戻る。m番地にはジャンプしないので注意。

例 EDSAC>HIPL GEROGERO /I

IPL ファイル名 [スイッチ1]……[スイッチn]

スイッチ
/B アドレス
……ブレイクポイントの設定
/F ファイル名
……ファイルへトレース結果を出力
/H……開始時刻,終了時刻の表示
/I……I命令読み込みデータの表示
/P……プリンタへ
トレース結果を出力する
/T……トレースを行う

機能 本物の初期入力ルーチンを実行する。すべての制御命令が使用できる。EmKPFを読み込めば当然m番地にジャンプする。

例 EDSAC>IPL TEST /B 100 /F /T /I

JUMP アドレス [スイッチ1]……[スイッチn]

スイッチ
/B アドレス
……ブレイクポイントの設定
/C……全レジスタのクリア
/F ファイル名
……ファイルへトレース結果を出力
/H……開始時刻,終了時刻の表示
/I……I命令読み込みデータの表示
/P……プリンタへトレース
結果を出力する
/T……トレースを行う

リスト1 n×n行列の掛け算

```
  1: PZ          /* 先頭
  2: T100K       /* 100番地から格納
  3: GK          /* 相対番地の基底は100番地
  4: K19#        /* サブルーチン先頭
  5: U23#
  6: U24#
  7: T25#        /* Accの内容をワークエリアに
  8: A23#
  9: T26#
 10: B6#
 11: F41#        /* サブルーチンコール
 12: A26#
 13: S22#
 14: G13#
 15: A21#
 16: F5#
 17: T27#
 18: A25#
 19: S22#
 20: G19#
 21: A21#
 22: F3#
 23: ZF          /* BbF命令に書き代わる
 24: FS2F        /* リターン
 25: P1F
 26: P2F
 27: ZF
 28: ZF
 29: ZF
 30: ZF
 31: ZF
 32: AF
 33: TF
 34: ZF
 35: ZF
 36: ZF
 37: HF
 38: VF
 39: PF
 40: BF
 41: P23#
 42: P24#
 43: P25#
 44: P26#
 45: K98#        /* サブルーチン先頭
 46: A23#
 47: T24#
 48: A39#
 49: A28#
 50: T101#
 51: A40#
 52: A28#
 53: T109#
 54: B50#
 55: F100#       /* サブルーチンコール
 56: T30#
 57: A24#
 58: T24#
 59: A38#
 60: A28#
 61: U109#
 62: T27#
 63: A39#
 64: A28#
 65: T101#
 66: B62#
 67: F100#       /* サブルーチンコール
 68: T31#
 69: A27#
 70: T101#
 71: A40#
 72: A28#
 73: T109#
 74: B70#
 75: F100#       /* サブルーチンコール
 76: T32#
 77: A31#
 78: A80F
 79: A33#
 80: T81#
 81: A32#
 82: A81F
 83: A34#
 84: T82#
 85: ZF
 86: ZF
 87: A35#
 88: T35#
 89: A24#
 90: S22#
 91: G90#
 92: A21#
 93: F54#
 94: T27#
 95: A30#
 96: A82F
 97: A29#
 98: T96#
 99: A35#
100: ZF
101: T35#
102: ZF
103: FS2F        /* リターン
104: K111#       /* サブルーチン先頭
105: ZF          /* 行列の要素のアドレスを求める
106: A36#
107: T104#
108: ZF
109: BS1S
110: A23#
111: J105#
112: S23#
113: ZF
114: S21#
115: ZF
116: FS2F        /* リターン
117: E400KPF
```

例 EDSAC>JUMP 100 /T /F OUT /B 110

OPEN ファイル名

機能 指定ファイルを入力専用としてオープンする。自作のプログラム中にI命令がある場合に，入力元として指定しておいてからJUMP命令で実行する。

REG スイッチ

スイッチ

/A……Accの値の変更

/B……Bレジスタの値の変更

/M……Mレジスタの値の変更

機能 全レジスタの表示を行う。任意のレジスタに任意の値を代入する。

レジスタと代入できる値の範囲

Acc……$-1 \leqq x < 1$

Bレジスタ……$0 \leqq x \leqq 1023$

Mレジスタ……$-1 \leqq x < 1$

SET アドレス

機能 指定したアドレスから命令または数値を格納する。"."（ピリオド）を入力するとコマンドラインに戻る。命令の入力は大文字でも小文字でも構わない。空白を入れても詰めても構わない。

例 EDSAC>SET 100

```
0100 P F   A200F (PFは以前の値)
0101 P F   a 201f
0102 P F   L   d
0103 P F   .(ピリオド)
EDSAC>
```

実際の使用とサンプルプログラム

とりあえず，サンプルプログラムをリスト1に示します（えどさっ君を使ってね)。このプログラムはn×nの行列の乗算を行うもので，全体としてサブルーチンとなっているので，メインルーチンとパラメータが必要です。まず，リスト1を打ち込んでください。ただし，「/*」以降のコメントは打ち込まないでください。便宜上コメントを付けただけです。

さて，このサブルーチンではAccにnを「PnF」で渡し（つまり，番地部にnが入っていて，ほかのビットはすべて0)，80, 81, 82番地にそれぞれ，行列1のデータ，行列2のデータ，結果を格納するアドレスの順に「PmF」のかたちで入れておきます。実際にやってみましょう。

まず，メインルーチンを用意します。

EDSAC>SET 400として，

```
A404F  /*  AccにP2Fを入れる
B401F  /*  この番地をBに
F100F  /*  サブルーチンコール
ZF     /*  終了
P  2F  /*  2*2の行列
.（ピリオド）
```

そして次に，

```
EDSAC>SET 80
 P300F  /*  行列1のアドレス
 P310F  /*  行列2のアドレス
 P320F  /*  結果格納先アドレス
```

最後に2つの行列のデータを入れます。

```
EDSAC>SET 300
 IF  /*   0.5
 RF  /*   0.25
 AF  /*  -0.25
 IF  /*   0.5
 .
EDSAC>SET 310
 ZF  /*   0.85
 AF  /*  -0.25
 WF  /*   0.125
 PF  /*   0.0
 .
```

サブルーチンを読み込ませて実行します。

EDSAC>IPL N_N.SRC(これはファイル名)

しばらくして，プロンプトに戻ったら，

EDSAC>DUMP 320

としてください。

```
320 U       F   0.4375
321 C       F  -0.125
322 BS512F     -0.140625
323 Q       F   0.0625
```

となっていればOKです。

おつかれさま

「よく頑張ったね」

ここまで読んでくれた読者に私は敬意を表します。これだけわけのわからない話が延々と続けばさすがにいやになったことでしょう。EDSACを知っていても，いまの世の中渡っていけるわけではありませんし，偉い人になれるわけでもありません。ただ，やはりノイマン大先生の考案されたアーキテクチャによる，世界最初のコンピュータであり，現在のコンピュータの祖先であるわけだから，墓参りをするような気持ちでEDSACを学んでほしい。そういった崇高なものなのです。ダサい（死語）とか，使ってらんねエとかいっちゃだめ。そうとなると，やはりEDSACのシミュレータも，崇高なものである。Z80や8086のシミュレータのようなトレンディな路線を狙ってるわけではありません。

「シブイね，えどさっ君」

「やるな，えどさっ君」

街でであったら，ぜひともこんなぐあいに声をかけてくれたまえ。じゃ。

micro computer 入門

マイクロプロセッサ潜入レポート

いまどきの32ビット高性能CPU

その昔，32ビットCPUは汎用機だけのものでした，しかし今ではホビー用パソコンにさえ搭載されています。32ビットは16ビットや8ビットとは何が違うのでしょうか？　少し難しいけど，この際，32ビットCPUについて勉強してみませんか。

Nakamori Akira
中森　章

32ビットCPUの特徴といったらなんでしょう。それはとりもなおさず性能です。32ビットとは「高性能」の別の表現といっていいかもしれません。

昔，32ビットといえば大型計算機のCPUのことを意味していました。大型計算機は非常に高性能ですが導入するためには何億というお金が必要で，小さな企業や個人にとって手の届く代物ではありませんでした。しかし，現在では100万トランジスタを1～2cm角のCPUチップに集積するほど半導体製造技術が進歩し，安価な32ビットマイクロプロセッサが製造されるようになりました。

現在，32ビットマイクロプロセッサ市場の伸びは著しく，従来大型計算機やミニコンが使用されていた分野は，非常に高性能が要求される分野は別として，すべて32ビットマイクロプロセッサに置き換わりつつあります。このような32ビットプロセッサもまた性能を上げるために大型計算機の手法を取り入れています。結局は規模の大小だけで32ビットという概念は同じものなのです。

やがて訪れるであろう犬も歩けば32ビットに当たるという状況に備えて，一般的な32ビットCPUというものの考え方に触れておくのも悪くない考えでしょう。ここでは32ビットCPUを理解するうえでの基礎となるマイクロプログラムとパイプラインの技法について説明しようと思います。ちょっと難しいかもしれませんが最新（でもないんだけど）技術の一端にでも触れるという気持ちで読んでみてください。

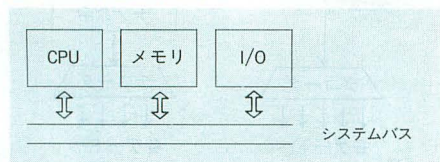



図1　マイコンシステム

図2　マイクロプログラムの概念

1　マイクロプログラム

図1を見てください。これはもっとも簡単なマイコンシステムのブロック図です。システムはCPU，メモリおよびI/O装置から構成されます。

CPUを動かすための機械語命令はメモリに格納されています。CPUは内部のプログラムカウンタの示すアドレスからメモリの内容（機械語命令）を読み込み，その命令の指示する処理を続けていきます。すなわち，メモリの内容を読み書きしたり，I/O装置との間でデータのやり取りを行います。

これらはちょっとマイコンをかじったことのある人ならば常識といっていいかもしれません。しかし，多くの人にとって機械語命令を与えられたときにCPUがどのようにして命令を実行しているのかを知っている人は少ないのではないでしょうか。まずはそれについて説明しましょう。

図3　CPUのブロック図

マイクロプログラムとは

今日の多くのCPUはマイクロプログラムと呼ばれる制御プログラムによって機械語命令を実行します。マイクロプログラムとはCPUの実行を制御する方法のひとつで，1951年にイギリスのケンブリッジ大学のWikes(バイケスと読むのかな)という人によって提案されました。

統計的にCPUの動作はいくつかの基本操作の組み合わせに分解できることが知られています。CPUに与えられる機械語命令をこの基本操作(これをマイクロ命令という)に分解して順番に実行してやろうというのがマイクロプログラムの基本的な考え方です。当然，機械語命令によって，たとえば加算命令と乗算命令では，行うべき処理がまったく異なりますから機械語命令ごとに別のマイクロ命令の組み合わせが存在することになります。

このようなマイクロプログラムはCPU内の適当な記憶装置(ROMであることが多く，マイクロプログラムROMあるいは単にマイクロROMと呼ばれる)に格納されていて，機械語命令が与えられると，それに対応するマイクロプログラムが読み出されて実行されるのです。図2にマイクロプログラムの概念を示します。このときマイクロプログラムの実行を制御するのはCPU内の制御回路です。図3にマイクロプログラム方式を用いるCPUのブロック図を示します。図3のブロック図を図1のマイコンシステムのブロック図と比べてみると非常によく似た構成になっているのがわかると思います。つまり，CPUの内部ではマイクロプログラムの制御回路がマイコンシステムにおけるCPUの役割を果たし，マイクロROMやレジスタがメモリの役割を果たしているのです。

マクロな視野から見たコンピュータシステムがミクロな視野から見たCPUの構造と似ていることは何か不思議な因縁を感じますね。そういうわけかどうかは知りませんが，

CPUに与える機械語命令をマイクロ(ミクロ)命令と対比してマクロ命令と呼ぶことがあります。また、マイクロプログラムをハードウェア(hard:非常に堅い)とソフトウェア(soft:柔らかい)の中間にあるものとしてファームウェア(firm:堅い)と呼ぶこともあります。「堅い」という語感を残してるあたり,マイクロプログラムはハードウェア寄りな概念だということができます。

マイクロプログラムについてもう少し詳しく見ていきましょう。マイクロ命令の1語には通常次のような制御情報が含まれています。

a) CPU内の回路を制御する情報
b) 次に実行するマイクロ命令を決めるアドレス情報
c) マイクロ命令で使用する定数

a)はマクロ命令でいうと転送命令や演算命令に相当し,具体的にはゲートを制御することによってレジスタや演算器といったCPU内のハードウェア資源に指示を与えたり,ハードウェア資源間でのデータ転送を行うための命令です。b)は分岐命令に相当します。c)の説明は不要でしょう。

マイクロ命令の種類

1語のマイクロ命令には以上で説明したような情報が符号化(エンコード)されて格納されているわけですが,この符号化の仕方によってマイクロ命令は水平型と垂直型の2つのタイプに分類することができます。

水平型マイクロ命令というのは,マイクロ命令の各ビットがCPU内の各ゲートに1対1で対応している命令です。この場合,制御すべきゲート数が多い場合はマイクロ命令の1語のビット長が大きくなり過ぎて実用的ではありません。通常はマイクロ命令を機能別にいくつかのフィールドに分割してエンコードし,実行時にそれらのフィールドをデコードすることで制御信号を作り出す方式が採用されます。マイクロ命令としては「ゲートAを開けろ」とか「ゲートBを閉めろ」というようにすべてのゲートに対する指示を記述することになります。つまり,1ステップですべてのゲートを制御することができます。水平型のマイクロ命令はCPUのハードウェアに非常に密着したものです。図4に水平型マイクロ命令を示します。

一方,垂直型マイクロ命令は同時に制御するゲートを制限することで,マイクロ命令の1語のビット数をさらに減少させています。垂直型マイクロ命令は原則的には操作コードとオペランドの組からなります。マイクロ命令としては「レジスタAとレジスタBを加えて結果をレジスタCに入れろ」といったぐあいにマクロ命令と非常に近い形式になります。

垂直型マイクロ命令ではゲートがどうこうといったハードウェア寄りな考慮は不要になります。この点,複雑なアルゴリズムを必要とするマクロ命令の実現には垂直型マイクロ命令が有効といえます。また,マクロ命令でのプログラミングと同様の手法を流用して効率的なプログラミングをすることもできます。ただし,垂直型マイクロ命令では1語のビット数を減少させたために,水平型マイクロ命令と同じ操作をしようとするとステップ数が増加する傾向にあります。これを解消するために,マイクロ命令の1語をいくつかのフィールドに分割し,それぞれのフィールドに垂直型マイクロ命令を記述する方法も考えられます。図5に垂直型マイクロ命令を示します。

一般的に,1語のビット数が大きく(100ビット以上)マイクロROMの容量が小さい場合は水平型マイクロ命令,1語のビット数が比較的小さく(数十ビット)マイクロROMの容量が大きい場合は垂直型マイクロ命令です。表1に水平型マイクロ命令と垂直型マイクロ命令の特徴をまとめておきましょう。

ナノプログラム

ところで,マイクロプログラムにもモジュール化という概念があります。マイクロ命令としてサブルーチンコール/リターンを用意することはよく行われます。しかし,現在の流行はメインプログラムに相当する処理をビット数の少ない垂直型マイクロ命令で記述し,サブルーチンに相当する処理をハードウェアに密着した水平型マイクロ命令で記述する方式です。

このような2レベルの構成を採用する場合,サブルーチン(水平型)はマイクロ命令よりも低い(ハードウェア寄り)レベルにあるとみなしてナノ命令と呼ばれます(マイクロは10^{-6}を意味し,ナノは10^{-9}を意味することは知ってますよね)。そしてナノ命令で記述されたプログラムをナノプログラムと呼びます。

この方式はともすれば複雑になりがちな垂直型マイクロ命令のデコードを簡略化し(命令数を減らすことができるから),処理の共通部分をひとつにまとめることができるためマイクロROMの容量を減少させることが期待できます。聞くところによると,X68000のCPUであるMC68000もこの2レベル・マイクロプログラム方式を採用しているということです。2レベル・マイクロプログラムの構成を図6に示しておきましょう。

マイクロプログラムの具体例

マイクロプログラムというものがどのようなものか見えてきたところでもう少し具体的なマイクロプログラムを説明します。

図4 水平型マイクロ命令

図5 垂直型マイクロ命令

図6 2レベル・マイクロプログラム

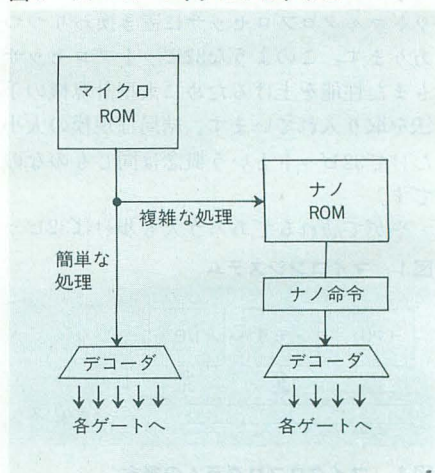



表1 水平型マイクロ命令と垂直型マイクロ命令

水平型マイクロ命令	垂直型マイクロ命令
1ビットが1ゲートを制御	同時に制御できるゲートを制限
1語のビット数はゲート数に比例	1語のビット数は比較的短い
小規模な回路に向く	複雑なアルゴリズム記述に向く
回路の制御性はよい	回路の細い制御には向かない
ROM容量は小さい 性能重視	ROM容量は大きい コスト重視

いま，CPU内の実行部の構成（演算器の周辺）が図7のような構成であると仮定しましょう。

図7のCPUはXバス，Yバス，Zバスという3種類の内部バスを持ち，これらのバスに定数1発生器，レジスタR0およびR1，ALU（演算器）がつながっています（ぶら下がっているともいう）。そして，ALUの片方の入力はXバス，もう片方の入力はYバスから行われ，演算結果はZバスに出力されます。ALUの演算はADD（加算）またはSUB（減算）を指定できるようになっています。そして，これらのハードウェア資源を制御するためのa～iという名前の9個のゲートがあります。

このようなCPUを制御するためのマイクロ命令を考えてみましょう。マイクロ命令は簡単のため水平型とします。ゲートが9個ありますから，マイクロ命令の語長は最低9ビット必要です。これにマイクロ命令の終了を示す1ビットを付加した全10ビットのマイクロ命令を考えればいいでしょう。図8がそのマイクロ命令で，各ビットが1対1にゲートa～iに対応しています。そして，マイクロ命令のビットが1である位置に対応するゲートが開くようになります。

それでは，以下の2つのマクロ命令を実現するマイクロプログラムを書いてみましょう。

1) レジスタR0の内容とレジスタR1の内容を加えてレジスタR1に入れる。

実現のためには次の操作が必要です。

・R0の値とR1の値をALUに入れるためにゲートdとgを開く。

・ゲートhを開けてALUにADDを指定する。

・ALUの出力をR1に入れるためにゲートbを開く。

したがってマイクロ命令はゲートb，d，g，hを開く指定をすればよく，

0	1	0	1	0	0	1	1	0	1

になります。この命令で実行は終了ですから，終了を示すビットも1にしてあります。

2) レジスタR0の内容とレジスタR1の内容を加えた値から1を引いてレジスタR1に入れる。

上の1）と同じ操作の後にR1の内容から1を引く操作を行います。このためには次の操作が必要です。

・定数1とR1の値をALUに入れるためにゲートcとgを開く。

・ゲートiを開けてALUにSUBを指定する。

・ALUの出力をR1に入れるためにゲートbを開く。

したがってマイクロ命令はゲートb，c，g，iを開く指定をすればよいことになります。このときマイクロ命令は2ステップ必要で，それは次のようになります。

0	1	0	1	0	0	1	1	0	0
0	1	1	0	0	0	1	0	1	1

上の1)や2)のマイクロプログラム例ではゲートが開くかどうかを考えてマイクロ命令の各ビットの0/1を決めました。しかし，実際のマイクロプログラム開発では人間の見やすい命令記述（ニーモニック記述）をマイクロ命令の0/1のパターンに変換するマイクロ命令用アセンブラが用意されるのが普通です。

このようなマイクロアセンブラがあれば，1)や2)のマイクロ命令は，たとえば，次のようなニーモニックをアセンブルするだけで作り出すことができます。

1)の場合：R1＝ADD(R0, R1)；END
2)の場合：R1＝ADD(R0, R1)…
　　　　　R1＝ADD(1, R1) ；END

命令デコードとの関係

ところで，これまで説明してきたマイクロプログラムは現実のCPUのマイクロプログラムと少し掛け離れています。先の例ではマクロ命令のオペランドがレジスタR0とR1に固定されていました。しかし，現実にはレジスタは16本程度あるのが普通です。

このとき，ある演算を行うマクロ命令のマイクロプログラムを，「R0の内容とR1の内容の演算」，「R0の内容とR2の内容の演算」，「R1の内容とR3の内容の演算」……というようにオペランドの組み合わせによって別個に作っていたのではステップ数の無駄になります。実際には「あるレジスタ」の内容と別の「あるレジスタ」の内容で「ある演算を行って「あるレジスタ」に書き込むというただひとつのマイクロプログラムで実行されます。そして，「あるレジスタ」というのが状況によってR0を示していたり，R1を示していたり，R2を示していたりするわけです。

また，「ある演算」は加算を示していたり減算を示していたりします。この場合のCPUの実行部の構成を図9に示します。この図でRXはXバスに値を出力するレジスタ番号を，RYはYバスに値を出力するレジスタ番号を，RZはZバスから値を入力するレジスタ番号を指示するレジスタです。ALUOPはALUで行う演算の種類を指示するレジスタです。

このCPUで演算を行うマクロ命令を実現するマイクロプログラムのニーモニックは，

図8　図7のCPUのマイクロ命令

0	1	2	3	4	5	6	7	8	9
aを制御	bを制御	cを制御	dを制御	eを制御	fを制御	gを制御	hを制御	iを制御	終了

図7　CPUの実行部

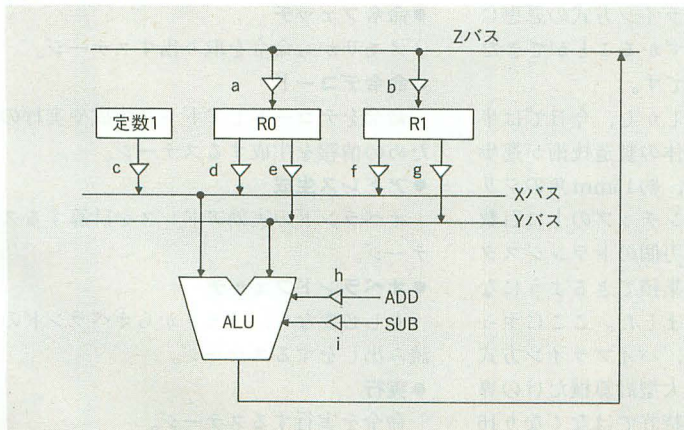
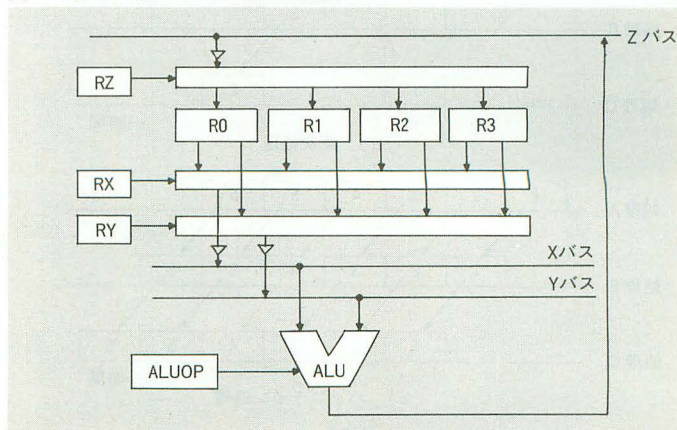



図9　より一般のCPUに近い実行部

たとえば，

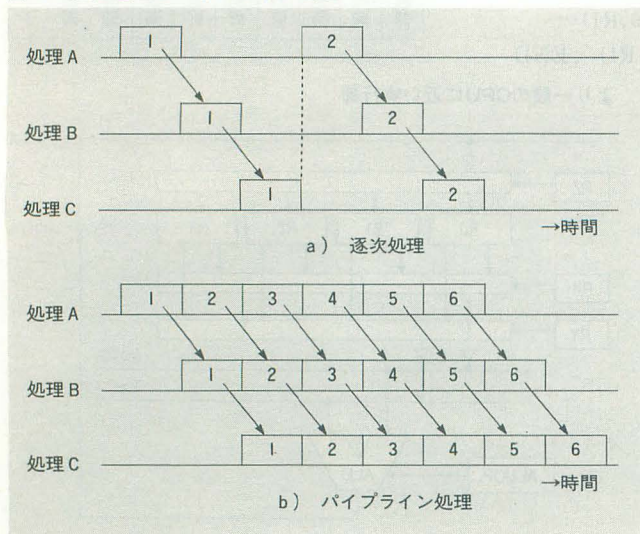
RZ＝ALUOP(RX, RY)；END

となります。このRX，RY，RZ，ALUOPはマクロ命令をデコードする時点で，マイクロプログラムの（マイクロROM内の）スタートアドレスと同時に作り出されます。逆にいえば，マクロ命令の命令コードからマイクロプログラムのスタートアドレス，RX，RY，RZ，ALUOPなどを決定する作業が命令デコードだということもできるでしょう。

マイクロプログラムの特徴

マイクロプログラム方式に対応する言葉がワイヤードロジック方式です。ワイヤードロジック方式はマイクロ命令を用いずにゲート間の配線のみによってマクロ命令の動作を実現する方法です。マイクロプログラムを用いるとマイクロROMの読み出し時間によって動作速度が制限されてしまいますし，マイクロ命令を逐次的に解釈して実行する性格上，すべてのハードウェア資源を並列に動作させることは困難です。つまり処理効果が落ちます。

このような理由からインテルの80486，モトローラの68040，NECのV80，富士通のGMICRO/300といった特に高速性を要求される最近の32ビットCPUではマイクロプログラムが使われなくなる傾向にあります。しかし，マイクロプログラムには次のような利点もあり，今後まったく用いられなくなるということはないと思われます。

1) CPUのアーキテクチャや命令セットの詳細が決まっていない段階でも回路設計を始めることができますから，設計期間を短縮できます。

2) 複雑な命令機能もマイクロ命令のステップ数を増やすだけで実現できます。そのための複雑な回路はあまり必要ではありません。

3) マイクロプログラムを変更するだけでCPUのアーキテクチャを簡単に変更できます。このため，特殊な目的のための専用チップ（カスタムチップ）の開発が容易にできます。

4) マイクロプログラムのバグを簡単に修正できます。もし，ワイヤードロジック方式を用いる場合，回路にバグを作り込んでしまったら修正が大変です。また，マイクロプログラムで回路上のバグを回避することも可能です。

2 パイプライン

今日のCPUでは非常に高い性能が要求されています。そのための一般的な方法がハードウェアを並列化して性能を上げるパイプライン方式です。パイプライン方式の発想は入出力処理（メモリアクセス）と演算処理をオーバーラップして行うことにあります。

これは1946年にはすでにアメリカのBurks（バークスかな？）によって提唱されていました。当時，メモリのスピードがCPUの内部動作に比べて非常に遅かったので，メモリアクセスが性能のボトルネックになっていたのです。

CPUの実行とメモリアクセスを並列に実行するための技術がパイプライン方式に発展していったわけですが，これを実現するためには非常に多くのハードウェア量が必要でした。このため従来は性能のためならコストに糸目をつけない大型計算機のみがパイプライン方式の恩恵にあずかることができたのです。

しかし，今日では半導体の製造技術が進歩し，約15mm角のシリコンチップの上に百数十万個のトランジスタを集積できるようになりました。ここに至って，パイプライン方式は大型計算機だけの専売特許ではなくなり16ビットのマイクロプロセッサでさえその技術を利用するようになっています。まして や，32ビットCPUでパイプライン処理を行わないものは考えられなくなってきているのです。

図10　逐次処理とパイプライン処理

パイプラインとは

パイプラインという言葉は製品が次々とパイプを通過して行く石油化学パイプラインに由来しています。これは最初の製品が最終工程から取り出される前に次の製品を投入し，流れ作業で処理を行っていくものです。これは自動車や電気製品の組み立てラインではもっともポピュラーな方法になっています。

図10にパイプライン処理の原理を示します。ある入力から出力を得るまでにA，B，Cといった3つの処理を行わなければならない場合，パイプライン処理を行えば通常の逐次処理の3倍の性能を達成することができます。これは出力が逐次処理の場合は3ステップに1回であるのに対して，パイプライン処理の場合は1ステップに1回であることを見ればわかるでしょう。このように3つの処理をパイプラインで行う場合を3ステージのパイプラインと呼びます。

いま，図10の横方向は時間の経過を表していて，縦方向はある時点で同時に行われる処理を示しています。これを見てわかるように，Nステージのパイプライン処理では同時に最大N個の処理を行うことができるのです。つまり，パイプラインのステージ数が大きいほど単位時間に行える処理が多いことになります。

さて，CPUでのパイプライン処理に話を移しましょう。CPUでの命令実行処理は命令を取り込んでからそれを実行して結果を書き出すまでがひとつの単位です。このときの処理は通常次の6つのステージに分かれています。

- **命令フェッチ**
  メモリから命令を取り出すステージ。
- **命令デコード**
  命令をデコードしアドレス生成や実行のための情報を生成するステージ。
- **アドレス生成**
  オペランドの実効アドレスを計算するステージ。
- **オペランドフェッチ**
  もし必要なら，メモリからオペランドの読み出しをするステージ。
- **実行**
  命令を実行するステージ。

●オペランドストア

もし必要なら，命令の実行結果をメモリに書き出すステージ。

ところで，パイプラインをスムーズに動かすためには各ステージの処理時間が同じになっていなければなりません。図11に各ステージの処理時間が１単位時間である場合と命令デコードステージのみが２単位時間で残りが１単位時間である場合のパイプライン動作を示します。

図11a)では命令は単位時間で実行されているように見えます。図11b)ではいちばん実行の遅い命令デコードステージの実行時間に引きずられて命令は２単位時間で実行されているように見えます

このようにパイプライン処理を行う場合の命令実行時間は一番処理時間の多いステージの処理時間になってしまいます。したがってCPUを設計する場合にはパイプラインの各ステージの処理時間を同じにする工夫が必要になってきます。逆にどれかのステージだけを高速に処理できるようにしてもあまり意味がありません。

実際のCPUでは，命令機能の複雑さに応じて，命令デコードステージと実行ステージの処理時間がばらついてしまいます。このためパイプラインステージの処理時間を一定にすることは並大抵のことではありません。これらを回避するための方法としては次のような方法が考えられます。

●命令デコードステージがばらつく場合

命令デコードステージ自体をさらに２段なり３段の，それぞれが単位時間で処理できるステージに分割してパイプライン処理をします。これによって，パイプラインのステージ数は増えることになりますが，パイプラインを単位時間でシフトしていくことができるため，単位時間での命令実行が可能になります。図11b)の命令デコードステージを，命令プリデコード（前処理）ステージと命令デコードステージに分けた場合のパイプライン処理を図12に示します。今度は単位時間で命令実行ができるようになりましたね。

●実行ステージがばらつく場合

これはほとんど手の施しようがありません。マイクロプログラムで実行制御を行っている場合はワイヤードロジック方式に変更して少しでも処理時間を減らす工夫をします。結局は，実行ステージがパイプライン処理のボトルネックになってしまうのです。また，実行ステージの処理時間が命令の実行時間ということになります。

ところで，現在広く出回っているCISC（Complex Instruction Set Computer）チップでのパイプライン処理の単位時間は少し前までは２クロックが基本でした。これはnMHzの周波数で動作させた場合（n/2）MIPSという性能になります。しかし，最近は１クロック処理が流行になり，その性能はRISC（Reduced Instruction Set Computer）チップに迫ろうとしています。この場合はnMHzの周波数で動作させるとnMIPSの性能を得ることができます。周波数nとしては25，33あるいは50MHzが流行ですから大変な性能ですね。

図11　パイプライン動作の比較

図12　デコードステージを２段に分けたパイプライン処理

命令フェッチ	1	2	3	4	5	6	7	8	9
命令プリデコード		1	2	3	4	5	6	7	8
命令デコード			1	2	3	4	5	6	7
アドレス生成				1	2	3	4	5	6
オペランドフェッチ					1	2	3	4	5
実行						1	2	3	4
オペランドストア							1	2	3

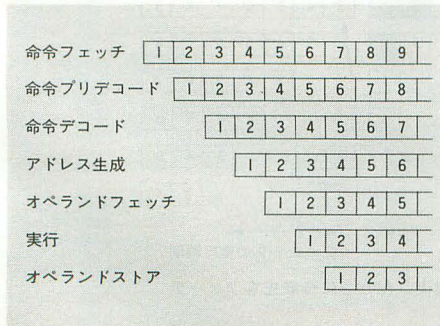



図13　パイプラインインタロックを生じる場合のパイプライン処理

ハザードとインタロック

パイプライン処理を行う場合，問題となるのがハザード（干渉）とインタロック（待ち）の問題です。たとえば，ある命令がレジスタの値をベースアドレス（レジスタの値がアドレス値となる）とするアドレッシングモードを持っていた場合，直前の命令がそのレジスタの値を変更するなら，その直前の命令の実行が終了するまでオペランドのアドレス生成ステージを開始することができません。このようなパイプライン処理ステージ間の干渉をハザードといいます。ハザードが出現したらそれが解消されるまでパイプライン処理の待ち合わせを行わなければなりません。この待ちをインタロックといいます。図13にインタロックを生じる場合のパイプライン処理を示します。

もしハザードが生じる場合，それに気づかずに（インタロックされずに）パイプライン処理が行われていったらプロセッサは誤動作してしまいます。このような誤動作をなくするために，パイプライン処理を行うプロセッサはスコアボーディングというハザード検出機能を持っています。これは命令の実行結果のストア場所（レジスタ）と１対１に対応するフラグビットを集めたスコアボードレジスタと呼ばれるレジスタに対して先行する命令が書き込み予約フラグを立て，後続する命令がそれを参照することで処理を開始できるか否かをチェックする機構です。

スコアボードレジスタの構成を図14に示します。要するに，アドレス生成ステージよりも先のステージ（オペランドフェッチステージ，実行ステージ，オペランドストアステージ）で処理されている命令が変更するレジスタの一覧表です。そして，ひとつの命令の処理が終了するたびにスコアボードレジスタはシフトされていきます。

さて，これまでの説明はアドレス生成に用いるレジスタを先行する命令が変更する場合（これをレジスタハザードという）でしたが，ハザードには次のようなものもあります。

●フラグハザード

条件分岐命令で参照する条件フラグを先行する命令が変更する場合。

●メモリハザード

命令のオペランドデータの一部または全部を先行する命令が書き換える場合。

これらのうち，レジスタハザードやフラグハザードはアドレス生成ステージで，メ

モリハザードはオペランドフェッチステージでインタロックが生じます。そして，その検出はスコアボーディング機構によって行われるのが普通です。

ところで，プッシュ命令，ポップ命令，関数コール命令あるいはリターン命令ではスタックポインタの更新が頻繁に行われます。これらの命令は通常のプログラムでよく用いられ，しかも連続して現れることが多いようです。しかし，どの命令もスタックポインタの値を使用してアドレス生成を行いますからレジスタハザードになってしまいます。このスタックポインタのハザードによるインタロックはCPUの全実行時間のうち相当な割合を占めることが予想されますから，これを回避することが性能向上のためには重要です。

このため，最近のCPUではスタックポインタを特別扱いし，ただスコアボーディングによってハザードを検出するだけではなく，値を予測してインタロックを回避しています。パイプライン処理の各実行ステージがスタックポインタのコピーを持っているのです。プッシュ命令，ポップ命令，関数コール命令，リターン命令ではスタックポインタの値がいくつ変化するかは一意に決まっていますから，命令デコード時にはアドレス生成時で使用するスタックポインタの値が計算できてしまいます。

特別な処理が行われるスタックポインタはともかく，図13から明らかなように，インタロックは命令の処理性能を低下させてしまいます。このため，パイプライン処理を行うプロセッサではインタロックが生じないようなプログラミングが必要になってきます。しかし，かしこいCPUになるとインタロックが生じる場合はその間を利用してインタロックとは無関係な後続する命令を実行してしまうこともあります。このような命令の追い越し処理は内部先行という名前で知られています。これはスコアボーディング機能を強化することで実現可能です。内部先行は大型計算機では昔から行われている技法ですが，現在のマイクロプロセッサではハードウェア量の関係からほとんど行われていないようです。

分岐命令の処理

パイプライン処理を乱す最も大きな要因は分岐命令です。分岐が生じるとそのときパイプラインの各ステージで処理されている命令はすべて無効になり，分岐先の命令フェッチからパイプライン処理をやり直さなければなりません。何も考えない実現方法では分岐先の命令フェッチは実行ステージの次にきます。このため，分岐命令の実行時間はn段のステージを持つパイプラインでは(n－1)単位時間になってしまいます。しかし，多くのCPUではアドレス生成ステージの次から分岐先の命令フェッチを行う（これを先行ジャンプ処理というらしい）ために3単位時間で分岐命令を実行することができます（実際はさらに1～2クロックのオーバーヘッドがあることが多い)。この様子を図15に示します。

以上は無条件分岐命令の例ですが，条件分岐命令となると，フラグハザードによるインタロックの問題もありますし，話がさらにややこしくなります。無条件分岐命令にしろ条件分岐命令にしろ，パイプライン処理を乱すことには違いありませんから効率的な分岐命令の処理が必要になってきます。ここではパイプライン処理において分岐命令を効率的に実行する手法について説明します。

●遅延分岐

分岐が生じた場合はその時点でのパイプライン処理のステージにある命令を無効化するのが原則です。しかし，図15をもう一度見てください。図15b)の場合，分岐先の命令フェッチによって無効化される命令3と命令4は分岐先の命令の実行を妨げることなく実行できるだけの空き時間が存在します。それならば，わざわざそれらの命令を無効化せずに実行しましょうというのが遅延分岐の考え方です。

遅延分岐を採用する場合，分岐命令の後のいくつかの命令が実行されてから分岐が行われることになります。遅延分岐に先立って実行が行われる命令の置かれている位置を遅延スロットといいます。

遅延スロットを活用すればパイプライン処理の乱れはなくなりますが，そこにどんな命令を持ってくるかが問題です。よほどプログラミング技術のある人か，非常にか

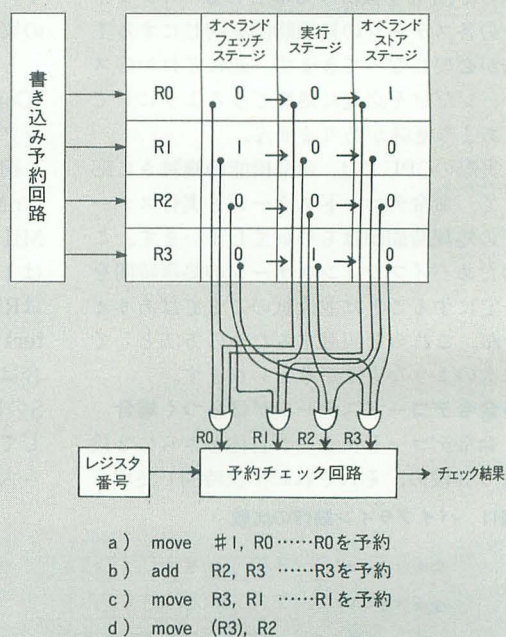
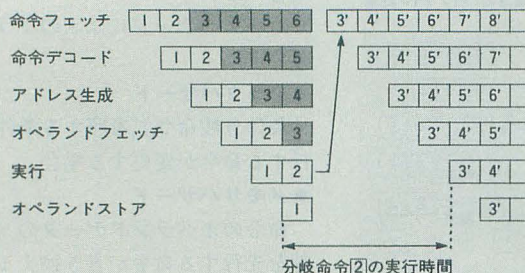
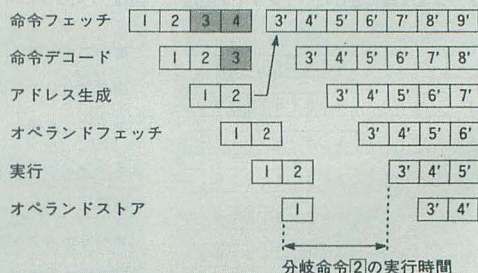



図14 スコアボードレジスタ

d)の命令がアドレス生成ステージにある場合のスコアボードレジスタ。 d)ではアドレス値としてR3を読み出そうとするが，R3は実行ステージにある命令 b)によって予約されている。

図15 分岐命令がある場合のパイプライン

a）何も考えない分岐処理

b）アドレス生成後，ただちに分岐先をフェッチ

しこいコンパイラでなければ効率的な命令実行は望めません。マイクロプログラムで使用されるマイクロ命令やRISCの命令では，命令の処理単位が単純化されているので，遅延スロットに置くと効果的な命令が多くあります。しかし，CISCのマクロ命令は多くの基本機能を集めたものですから，分岐のついでにちょっとというような命令はあまりありません。このため，CISCのCPUでは遅延分岐が採用されることはないようです。マイクロ命令やRISC以外には，DSPが遅延分岐を採用していることが多いようです。

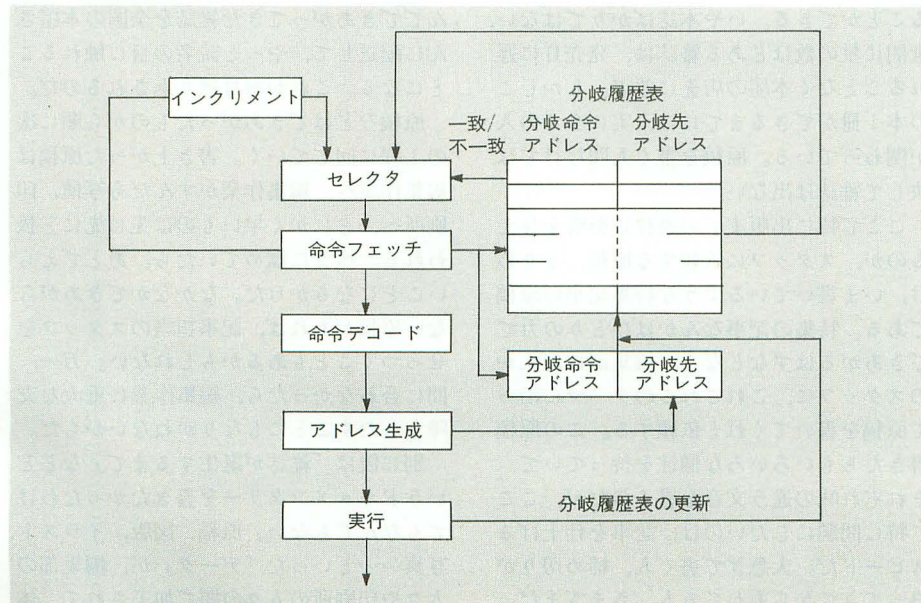
●分岐予測

条件分岐命令の分岐先を予測して命令フェッチを行う技法が分岐予測です。条件分岐命令があるとCPUはそれが分岐するか分岐しないかを予測して，分岐した方向に命令フェッチを行います。この場合，パイプラインの各ステージの処理結果にはそれが仮の結果であることを示すタグ（目印）が付けられます。そして，最終的に分岐が決定されて予測と一致していれば，タグは取り除かれます。もし一致しないならば，各ステージで現在処理中の命令を無効化して命令フェッチから処理をやり直します。

ここで，ある分岐命令が分岐するか否かの判断は，分岐命令の種類やコンパイラが生成する命令列の性質が参考になります。たとえば，ループ命令は分岐する確率が高いといえるでしょう。あるいは，

if（条件）then 処理A else 処理B

という高級言語の1文に対して，

```
TEST  条件  ; 条件のテスト
BEQ   処理B; 不成立なら処理Bへ
```

というコードを生成するコンパイラにおいては，BEQ(Branch if equal）という条件分岐命令は分岐しない確率が高いといえます。なぜなら通常パス（処理する確率が高いほう）はelseよりもthenの後に書かれることが多いと考えられるからです。最近ではCPUの開発とコンパイラの開発は同一メーカーで行われることが多いのでコンパイラの性質に合わせたチューニングは難しいことではありません。

●分岐履歴表とデコード履歴表

分岐予測によってよい性能を得るためには，分岐するか否かの予測を高い確度で行わなければなりません。その予測を過去の振る舞いに従って行うのが分岐履歴表を用いた分岐予測です。分岐履歴表とは，同じ分岐命令が実行されるときに正確な予測を行うための情報を格納しておく表のことです。

分岐予測のもっとも単純なやりかたは，前回の分岐と同じ方向に分岐すると予測することです。このときの分岐履歴表は，分岐命令のアドレスと分岐先のアドレスのペアを覚えておく一種のキャッシュメモリになります。図16に分岐履歴表を実現する方法のひとつを示します。

分岐履歴表は命令フェッチのたびに参照されます。もし命令フェッチアドレスが分岐履歴表の分岐命令アドレスと一致すれば，以後の命令フェッチはそれに対応する分岐先アドレスから行われます。もし一致しなければ分岐命令の命令長を加えた次のアドレスから命令フェッチが行われます。そして，実際に分岐命令が実行ステージを通過した後に，分岐命令アドレスと正しい分岐先アドレスで分岐履歴表を更新します。

分岐履歴表は確かに効果は期待できますが実現するためにハードウェア量が多くなるのが欠点です。それに対してもっと簡単な方法で分岐する方向を予測する方法が考えられています。それがデコード履歴表です。これは分岐履歴表とは異なり，分岐先アドレスを記憶しておくのではなく，分岐命令アドレスに対して分岐するか否かを示す1ビットを記憶しておく方式です。この1ビットは分岐命令のアドレス生成ステージで参照し，次に行う命令フェッチの方向を予測するものです。

分岐履歴表は命令フェッチの段階で分岐先を予測しますから，予測が成功した場合，分岐命令の実行は1単位時間でできます。それに対してデコード履歴表では図15b)の無条件分岐の場合と同じで3単位時間で分岐命令の実行を行います。分岐予測は大型計算機では一般的な技法ですが，マイクロプロセッサには最近になってやっと採用されるようになりました。NECのV80と三菱電機のGMICRO/100が代表的なところでしょう。V80は分岐履歴表の考え方を利用し，GMICRO/100はデコード履歴表の考え方を利用した分岐予測を行っているようです。

＊

32ビットCPUを考えるときキーポイントとなるマイクロプログラムとパイプラインの技法について簡単に説明してきましたがどうだったでしょう。やはり，ちょっと難しかったかもしれませんね（パソコン雑誌にこんな記事を書くほうも書くほうだがそれを載せるOh!Xも困ったものだ）。

プロセッサの性能を向上させるためにはパイプライン処理を効率よく実行させることが重要です。命令実行のワイヤードロジック化や分岐予測機構なども各パイプラインステージの実行時間を均一化するための一手法であると考えればすっきりするのではないでしょうか。CPUのアーキテクチャもこのようにひとつの観点から見ていくとなかなかおもしろいものです。

図16 分岐履歴表

◆参考文献

1）萩原宏，「マイクロプログラミング」，産業図書，1977年。

2）ハロルド・S・ストーン，「高性能コンピュータアーキテクチャ」，丸善，1989年。

3）金子博明ほか，「キャッシュと分岐予測機構の内蔵などでパイプラインの乱れを抑えて性能を上げた32ビット・マイクロプロセッサV80」，日経エレクトロニクス，1989年6月26日号(no. 476)，pp. 141-152。

4）吉田豊彦ほか，「先行ジャンプ処理の採用によりパイプライン処理効率を高めた32ビット・マイクロプロセッサGMICRO/100」，日経エレクトロニクス，1989年7月10日号(no.477)，pp. 185-196。

新しいアーキテクチャを見る

ヘンなコンピュータ

データフロー・マシンやニューロコンピュータなど，いま話題のちょっとヘンなコンピュータについてわかりやすく解説します。ヘンなコンピュータと現在のコンピュータが形成するコンピュータ社会の未来像を見ていきましょう。

Tan Akihiko
丹　明彦

人は計算機になにを期待してきたか，またなにを期待しているのだろうか。

人間の代わりになんでもやってくれること，人間を雑事から解放すること……，こんなところであろう。かつての僕もそうだったのだが，コンピュータに触れたことのない人は，コンピュータといえば万能の機械，どんな難問もたちどころに解決してくれるという幻想を持っているものだ。

よくご存じのとおり，この楽観的な期待とは裏腹にコンピュータは単なる計算機，電卓の親玉としての存在でしかなかった。そう，誤解してはいけない。コンピュータは決して「なんでもできる」のではない。人間が手順をきっちりと教えさえすれば，人間が手作業で計算する代わりに高速かつ正確に計算する，それがコンピュータだ。

逆にいえば，原始的な計算機言語の範囲で表現できない作業に対しては，コンピュータはまったくのダルマさんになってしまう。この「コンピュータ＝道具」論を踏み誤るから，「高価なパソコンを買ったはいいが，いまはホコリをかぶってしまっている」というパターンの話があとをたたなくなってしまう。コンピュータに期待しすぎてはいけない。過度の期待に応えるには，いまの計算機というのは少しばかり無愛想すぎるのだ（無愛想な職人に腕の立つ者が多いが，コンピュータもこと数値計算に関してなら右に出る者はいまい)。

さて，そんなこんなで「コンピュータ＝電子頭脳(古い)」という幻想は崩れていき，コンピュータは「電算機」という呼ばれ方をされるようになった。こうなるといよいよ「理系バリバリの人間が使う計算専用機」のイメージが定着したようで，「私は文系人間だからコンピュータは全然受け付けない」的発言もちらほら。個人的には，文系の人にもどしどし使ってもらったほうが，コンピュータ界の発展のためにはいいと思うのだが。計算機は次第次第に怪物化していって，コンピュータ嫌いの人からますます遠い存在になってしまいそうだ。

誤解を招くとぐあいが悪いので断っておくが，僕はそのこと自体が決して悪いといっているのではない。いい悪いではなく，コンピュータの成長がこういう方向にきたというだけのことだ。もちろん，専門家が使う限りは強力なのがいいに決まっている。コンピュータの高性能化は歓迎すべきだ。

しかし毎日のようにコンピュータを扱っていると，客観的に見た「いまの」コンピュータの姿を見失ってしまうのではないか。さらに，望まれている「これからの」コンピュータの姿も見えなくなってしまうのではないか。確かにアーキテクチャがどうのこうのとかいう議論は僕にとって大変興味深いところだ。しかし，こうした僕らが日常接しているコンピュータからかけ離れたコンピュータについて考えることで，逆に現在のコンピュータの抱える問題を見つめ直す機会ができるかもしれない。川の流れに流されるだけではなく，ときには岸に上って流れを見つめようではないか。

データフロー・マシン

僕らは毎月このOh!X誌を本屋で手にすることができる。いや本誌ばかりではない，世間に星の数ほどある雑誌は，発売日に遅れることなく本屋の店先に並ぶ。しかしこの本1冊ができるまでには，実に多くの人が関わっている。原稿を書く人間だけでは決して雑誌は出ない。

ここで特に出版までの過程に影響を与えるのが，スタッフに依頼する原稿，とりわけ，いま書いているような特集記事の原稿である。特集の記事なんかはひとりの力でできあがるはずなどなく，編集部は何人かのスタッフに，これこれこのテーマに沿って原稿を書いてくれと依頼する。この原稿書きたちもいろいろな個性を持っていて，それぞれ味の違う文章を書くのだが，ここで特に問題にしたいのは，記事を仕上げるスピードだ。大急ぎで書く人，締め切りが迫ってきてからあわてる人，さまざまだ。

だから仕上がり日が人によって違うのは至極当然で，締め切り日は同じはずなのに原稿の入りぐあいが違うからといって，すべてが筆者たちの責任とは限らない。

で，早くできあがった原稿から順に，編集作業が始まる。内容が適切でないところは手直しをされ，もちろん誤字脱字もチェックされる。図版の指示やリストの出力，連載記事などにはイラストも入れてもらうし，ゲームレビューには写真の手配も必要だ。材料が揃ったらレイアウターに渡され，見栄えがするようにレイアウトされる。

そのレイアウトに従って図は清書され，文字は写植で綺麗に仕上げられる。写植屋さんから初校が上がってくるので，赤いペン片手に校正が始まる。誤字があったらもちろん片端からチェックしていく。これは主に校正屋さんの仕事だ。さらに技術スタッフなどにより内容の再チェックが行われ，最終的に編集で校正がまとめられる。

校正がすんだら，写植屋で訂正がおこなわれ，ここで初めて印刷屋さんに回す。写真の製版などができたところで，今度は印刷所に直接出向いていって最後の校正を行い，正式に印刷が始まる。印刷・製本がすんでできあがってきた雑誌を全国の本屋さんに配送して，やっと読者の目に触れることになる。これが毎月繰り返されるのだ。

原稿などはできあがったものから順に次の工程に回していく。書き上がった原稿は編集作業へ，編集作業がすんだら写植，印刷所へ，とにかく早いものは先に先にと扱われる。ヘタに溜めていたら，あとでえらいことになるからだ。なかなかできあがらない原稿があれば，記事担当のスタッフをせっつくこともあるかもしれない。万一，間に合わなかったら，編集作業に重大な支障をきたすことにもなりかねないからだ。

別に僕は「雑誌が誕生するまで」などというドキュメンタリーを書きたかったわけでもなんでもない。原稿，図版，イラスト，写真……といった「データ」が，編集部の人々や印刷所の人々の間で加工されて，体

裁よく1冊の雑誌という「結果」にまとまっていくようすを比較的わかりやすい例でいったまでだ。

さて，ここで計算機に話題を戻そう。計算機は「データ」を飲み込んで，ある規則に従って加工し，「結果」を吐き出す機械である。走るソフトウェアによってデータの形態も処理内容も違うので，見かけ上はまったく違っても本質的には同じだと思う。コンパイラだろうがゲームだろうが，多分この法則に当てはまるであろう。

現在主流となっているノイマン型のコンピュータは，メモリの上に行儀よく並んだ命令語を順番に処理する。プログラミング用の言語を使って処理の手順を具体的に記述するのだが，ノイマン型コンピュータでは，処理する順番に，数式などを記述していけばよい。これは，シングルプロセッサを搭載するコンピュータにとって，「ひとつの」理想的な形態だと思う。

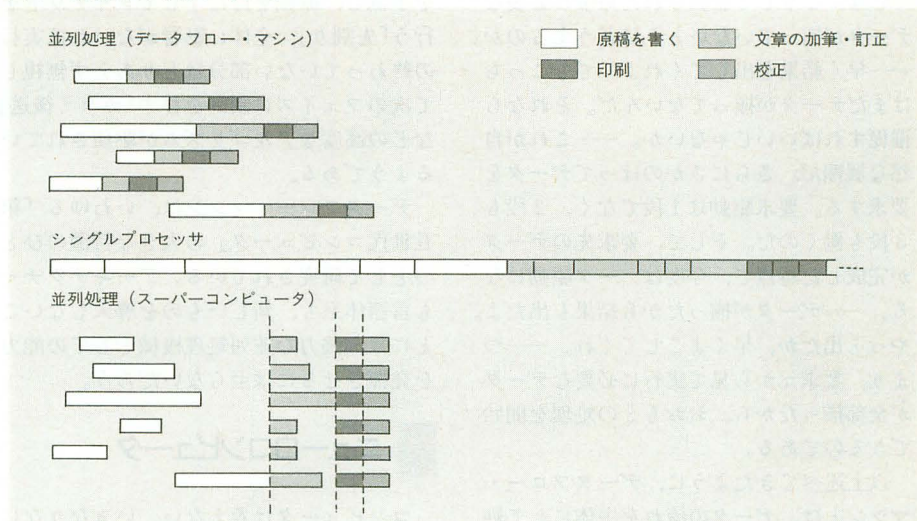
一方，雑誌出版のほうはどうか。まず原稿書きからして何人もいる。編集部の人々なども含めて，たくさんの人々が関わっていることは前に書いたとおり。コンピュータにたとえるとマルチプロセッサということになるだろう。さすればこの作業は「並列処理」で行われていることになる。

嘘だと思うなら，シングルプロセッサ，すなわちたったひとりで雑誌を出すことを考えてみよう。原稿を書くのも当然ひとり。全部あわせて100ページ余りにもなる文章をひとりで書き続けることになる。図版も写真もイラストも，すべてひとりでこなしていく。印刷所にいって製版をするのも自分なら，校正をするのも自分……1カ月でできるわけがないことはおわかりいただけることと思う。並列処理は処理を効率よく終わらせるための必須アイテムなのである。

さて，原稿その他が「データ」であることはすでに述べたが，そのデータの「流れ」をちょっと見てほしい。編集者はスタッフに原稿を依頼し，できあがってくるのを待つ。仕上がりが遅ければせっつく。原稿ができて初めて次の作業にかかれる。

これをコンピュータ的にいいなおそう。プロセッサAはデータを用意しろとプロセッサBに要求し，プロセッサBはデータの作成を始める。データが完成した時点でプロセッサBはプロセッサAにデータを送り，データを受け取ったプロセッサAはこのデータを加工する作業を始める。

少しわかりにくくなった。いきなり本題に入ろう。ここに「データフロー・マシン」の真髄があるのだ。データフロー・マシンのキモは次の2つである。

図1　雑誌の作り方いろいろ

●データ駆動

ある命令を実行するときには，その命令に必要なデータが揃うまで待つ。データが揃ったらその命令をいつでも実行できる。

これは，「データの流れ」が命令を駆動し，作業の進行を制御しているということを意味する。ノイマン型コンピュータでは，処理を制御するのは決してデータではなく，プログラムカウンタであった。データでなく，プログラム主導型なのである。メモリ空間上に並んだ命令を順番に実行しているこの方式で，「データが揃うまで待つ」などということをやっていれば，プログラムの進行はそこでストップしてしまう。

たいがいのプログラムではデータの流れは1本ではない。たくさんの細い支流が集まって大きな川になるように，また数ページの原稿を集めていって100ページ余りの雑誌ができるように，データというものは，要求する側も供給する側もひとつ（ひとり）ではなくいくつも（何人も）あるのが当たり前だ。ノイマン型では，供給するデータを1つひとつ用意し，それを要求する命令に1つひとつ回していかなくてはならない。だから，処理中でないデータは，まるっきり遊んでいることになり，プログラムカウンタが自分のところにやってくるのをただただ待ち続けるという構図が成立する。

データ駆動は，複数のプロセッサが独立にデータを処理しているからこそ可能な技なのである。

なお，ノイマン型コンピュータでも，並列処理を行っているものがある。スーパーコンピュータがそれである。しかしスーパーコンピュータの並列処理（カコミ参照）は，データフロー・マシンのそれと比べるとやや不自然である。

スーパーコンピュータのどこが不自然なのか。並列処理といっても，ループ文を一括して処理しているだけのことで，結局はプログラムカウンタの呪縛からは逃れきっていないのである。アレイプロセッサはそれぞれ独立に動作してはいるが，プログラムカウンタとは独立でない。ループを処理しているあいだ，全体の処理の進行は止まってしまっている。プロセッサはここにかかりきりになっているのだ。これを雑誌出版とのアナロジーでいえばどうか。いささか強引だが，次のように考えてみた。

スタッフは一斉に原稿を書き始める。編集諸氏は全員分の原稿が出揃うのを待っている。仮にひとりが異常に遅らせても，平気で待ち続けている。編集待ちの原稿がちゃんとあるのに，編集部が働いていないのである。やっと全員分揃ったところで，編集作業に入る。以下，校正も印刷も，全員分の処理が終わるまで次の行程には進めない。シングルプロセッサ同様，この方式でも1カ月で雑誌が完成するかどうか怪しいものがある。

●要求駆動

ある命令の実行結果が必要になったときに，その命令を駆動する。

たとえば週刊誌などで，連載陣のひとりが急病で休載し，穴埋めに新人の作品が載ることがたまにある。これは「要求駆動」にちょっと近いと思う。仮に，休載によってページに穴があかない限り，この新人さんの出番はなかったとしよう。これは，実行結果が要求されない限り命令が駆動されないという，要求駆動の特性と通じる。要求駆動では必要のない処理はしなくてすむので，計算時間に無駄がなくなる。

では，こうして駆動された命令に必要なデータが揃っていないときはどうするのか。――早く結果を出してくれよ。でもこっちはまだデータが揃ってないんだ。それなら催促すればいいじゃないか。――これが自然な展開だ。さらにさかのぼってデータを要求する。要求駆動は1段でなく，2段も3段も働くのだ。そして，要求先のデータが完成した時点で，今度はデータ駆動になる。――データが揃ったから結果も出たよ。やっと出たか，早くよこしてくれ。――つまり，要求元から見て実行に必要なデータが全部揃ったから，おおもとの処理を開始できるのである。

以上述べてきたように，データフロー・マシンとは，データの流れを主体にして処理が進行する処理系で，並列処理に向いた処理方法だということができる。しかし，すべての処理が並列に処理してかまわないということは稀である。前の処理の出力がないと処理が滞ることが多いのだ。並列処理を効率よくするためにはこういったことにも対処しなければならない。最先端の編集現場では，すでに一度実行が終わっている部分は処理を省略する「前号流用」や通常処理の因果関係に逆らって原稿の内容を予測し，原稿を待たずしてレイアウトを行う「先割り」，全体に影響がなければ実行の終わっていない部分はとりあえず無視して次のフェイズに制御を移す「カコミ後送」などの高度なアルゴリズムが駆使されているようである。

データフロー・マシンは，いわゆる「第五世代コンピュータ」の大きな特徴のひとつとして研究されている。アーキテクチャも言語体系も，新しいものを導入しないことには，強力な並列処理機械としての能力を発揮させるには至らないだろう。

ニューロコンピュータ

コンピュータは考えない。いきなりなにをいうのかと思った方も多かろう。コンピュータはその名のとおり単なる計算機，数値処理機なのだ。確かに長時間計算させるとコンピュータは「考えて」いるように見える。が，内部で行っているのは膨大な量の算術/論理演算，それにデータ転送くらいのもの。これをいつ果てるともなく繰り返しているにすぎないのだ。コンピュータと脳のアナロジーがいわれて久しいが，両者は全然似ていないといっていいくらいだ。

コンピュータはほんの少しの誤りにも弱い。BASICプログラムで，つまらないタイプミスのために「SYNTAX ERROR」攻撃を食らって腹を立てなかった人は多分いないだろう（もちろんタイプミスをしたほうが悪いのだが，人間はそこを素直に反省できない生き物なのだ。少なくとも僕はそうだった)。コンピュータのおかげで人間はこのうえない計算の正確さを得たのだが，その代わりに使う側にも絶対の正確さ，細心の注意力を要するようになったのである。

このことと少し関連するが，コンピュータは故障にも弱い。システムの一部が死んだだけでも，それは全体にとって致命的である。ある大型コンピュータの話。この計算機に使われている素子が多かった。そのうえ，素子の故障率もけっこう高かった。さてどうなるか。数分から数時間稼働するともうシステムがダウンする。それを修理してまた走らせる。あっという間にまたダウンする。これでは動作が超高速だとしても，全体の速度としては疑問である。

なんだかコンピュータの悪口ばかり書いてしまったようだ。僕はこれらのことを欠点だとはなるべく思いたくない。むしろ計算機の持つ強烈な個性というふうにとらえたい。しかし，コンピュータが普及するうえで，これらの個性は確実にマイナスの要因として働くであろう。コンピュータが使いにくいと思われている原因の多くがこの辺にあるからだ。

これらのことは，コンピュータが「曖昧性」を持たないことに端を発している。逆に，人間の脳は，実にこの曖昧さのおかげで，「ある方面に関しては」コンピュータに比べてはるかに優れた働きをする。コンピュータと脳は，守備範囲がまったく違うのだ。数値計算の能力に限ればコンピュータに遠く及ばない人間の脳も，推論，判断，認識といった処理をさせればはるかに優れている。両者がお互いに補完し合えばうまくいきそうだ。

そこで出てくるのが，頭脳の働きを素直に真似てみようではないかという発想である。生物の身体は，我々人間の獲得してきた文明などではとても解明できそうにない巧妙な仕組みで働いている。頭脳の働きをモデルにした，まったく新しい（発想自体は古くからあったのだが）コンピュータがいま話題を呼んでいる。それがニューロコンピュータである。

似たものにバイオコンピュータがあるが，こちらは生体素子を使うコンピュータ。ニューロコンピュータは，特に生体素子にこ

スーパーコンピュータの並列処理

最速のコンピュータ，スーパーコンピュータ。この処理系ではFORTRANが動く。さて，一般的なFORTRANプログラムでは，処理時間の大部分を占めるのが繰り返し演算である。そこに目を付けたコンパイラ設計者は，FORTRAN最適化の一環として，繰り返し処理の「自動ベクトル化」を考えた。

FORTRANのループは，ご存じのとおり（？）DO文である。BASICではFOR～NEXT文にあたる。

```
      DO 10 I=1,N
   10 C(I)=A(I)+B(I)
```

というプログラムがあった場合，ふつうは配列の中身をひとつひとつアクセスしていく。A(0)とB(0)を足してC(0)に入れる，A(1)とB(1)を足してC(1)に入れる，……，A(N)とB(N)を足してC(N)に入れる，となる。これは決して効率がよいとはいえない。そこでスーパーコンピュータは，ベクトルレジスタと呼ばれるレジスタを用意した。ふつうのレジスタは値をひとつしか格納できないが，ベクトルレジスタでは同時に64個とか256個とかいったぐあいに複数の値を同時に（1命令サイクルで）格納・処理できる。当然，それに対応して演算器も64個，256個用意されている（これをアレイプロセッサと呼んでいる)。そしてFORTRANコンパイラは，ループ文が出てくると，できる限りベクトルレジスタで処理させようとする。上のプログラムはもっとも簡単な例で，Aベクトルレジスタに配列A(I)の内容を，BベクトルレジスタにB(I)を，CベクトルレジスタにC(I)を対応させる。そして“C=A+B”の命令一発で，配列の中身が一気に足される。アレイプロセッサがN個の加算を同時に行うので，処理時間は(理論上)足し算1回分ですみ，N倍の高速化ができた勘定になる。

a)プロセッサとレジスタの構成

b)N組の数の加算

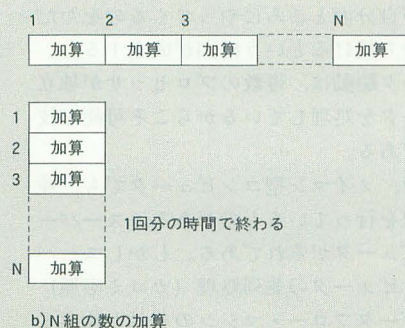
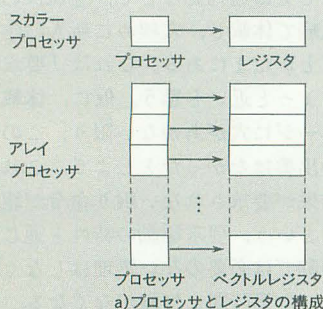
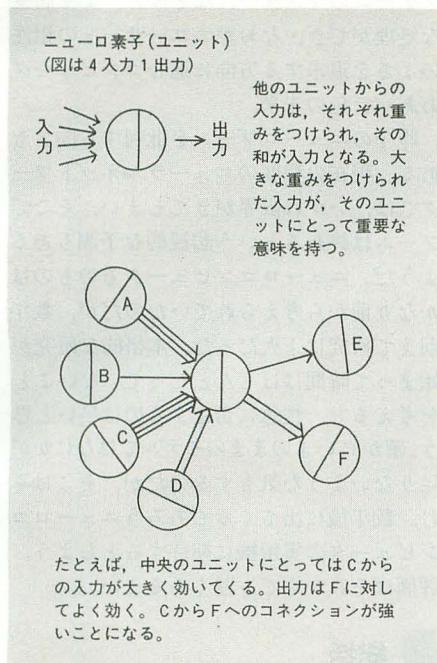



図２　ニューロ素子

だわらなくとも，脳の働きに近いコンピュータを指す。ちなみにニューロとは神経細胞のこと。神経の働きにモデルを求めているのだ。

いま盛んに研究されているニューロコンピュータの構造はニューラルネットワークと呼ばれている。ニューラルネットには大きく分けて図に示す２つのモデルがあるが，もちろんまだ研究途上で，もっといいモデルが出てくる可能性もある。どちらのモデルにも共通しているのは，たくさんのニューロ素子があって，そのニューロ素子間に結合を作っている点である。この結合というのは，信号を送る線のようなもので，この結合の強さで物事を表現する，というしくみになっている。

ニューロコンピュータは学習によって目的を達成する。もちろん目的に応じて学習の方法を変える。繰り返して学習するうちに，次第に高い率で正解が出せるようになるのである。通常のコンピュータのように，あらかじめプログラムされた論理的な処理で答えを出すのとはまったく違う。むしろ，連想で答えを出すといういい方が正しい。たとえば誰でも掛け算の九九を覚えたとは思うが，まさに理屈抜きで覚えたと思う。九九が口をついてスラスラ出るのは，反復的学習の成果である。

九九に限らず，人間の行為というものは実のところみんなこうで，たとえば自動車の運転にしても，操作の手順を考えながら運転している人は免許取りたての人だけである。これも練習するうちに，動作を身体が覚える。つまり脳の中にそういう神経のパターンが根づいてしまうのだ。この神経のパターンが，すなわちニューラルネットワークの結合の重みに対応する。ニューロコンピュータが学習を終えたとき，その内部には，入力から出力まで滑らかにつながる結合のパターンができていることだろう。よく使う水路は太くなっていき，めったに使わない水路は枯れていく。これと同じことで，判断に重要でない結合は弱く，重要な結合は強く――学習を重ねるにつれて，この状態に少しずつ近づき，正解率が高くなる。この意味で，ニューロコンピュータは論理でなく連想で答えを出すというのである。

人間の脳は構造そのものが記憶内容と密接に関わりを持っていて，記憶と構造が切り離して考えられない。ニューロコンピュータで「学習」とは，つまるところ，答えを出すための最適な構造を作り上げていく作業なのである。

学習にはときどき間違いも起こる。九九を１回で正確に覚えた人もいるだろうが，６×７だとか，そのあたりで一度くらい間違えて覚えた人はいないだろうか。頭の中に間違った水路が引かれてしまったのだ。このままでは算数の成績が危ない。間違った水路を埋め立て，正しい水路を引き直す必要がある。ここで，フィードバックが登場するのだ。６×７＝48。違う。42だ。ろくしちしじゅうに，ろくしちしじゅうに……。間違った連想を矯正するために，正しい答えを繰り返し暗唱して，新しい水路を引く。この，出力「48」が間違いという結果を入力「６×７」の段階に戻してやって，学習をやり直すことをフィードバックという。ニューロコンピュータの多くは，学習過程にフィードバックを取り入れている。

ニューロコンピュータには，基本的に「プログラム」がない。試行→結果の評価をフィードバック→再試行，の繰り返しで学習するだけ。逆にいえば，学習させるだけでどんな処理もこなせるようになる可能性があるわけだ。人にモノを教えるのに，特殊な装置が必要だろうか。

データフロー・マシンは並列処理で効率よく処理を行うシステムだったが，ニューロコンピュータもれっきとした並列処理である。現在はノイマン型コンピュータでニューロコンピュータのシミュレーションも行われているようだが，並列処理という部分がネックになっていて，専用のニューロチップには速度の点でも遠く及ばない。

ニューロコンピュータが学習する例をあげてみる。通常のコンピュータと似ても似つかない使い方をしていることがおわかりいただけると思う。ニューロコンピュータの典型的な利用法なのだが，入力した画像の文字を読み取る，いわゆる文字認識。

従来の（ノイマン型コンピュータでの）パターン認識は，複雑な判定用のプログラムを必要としていたし（もちろんそのために死ぬ思いで頭を絞るのはプログラマである），それでも正確に読み取るのは至難の技であった。ここにニューロコンピュータを導入すると，事情が変わってくる。準備するのはニューラルネットワークのみ（もちろん，文字を読ませるための画像入力装置なども必要だが，ここで強調したいのは「プログラムが不要」だという点である）。初めニューラルネットは適当な初期状態になっている。つまり，各ユニット間の結合状態（結合の強さ）は決まっていない。まったくランダムなこともある。これからの学習によって，だんだん最適な文字判別用のネットワークに育っていくのである。

学習サンプルを入力しながら，それに対応する出力を教え込む。たとえば“A”という文字を画像入力してそれをネットワークに通す。オペレータは「こういう入力があったら“A”という出力を出せ」とオペレータがネットワークに指示する。たとえば，“A”を画像入力して，“A”のキーを押すのである。これを「教師信号」と呼んでいる。人間だって誰かに教わり，練習しながら物事を覚えていく。教わる段階ならば指導も必要だろう。

図３　代表的ネットワーク

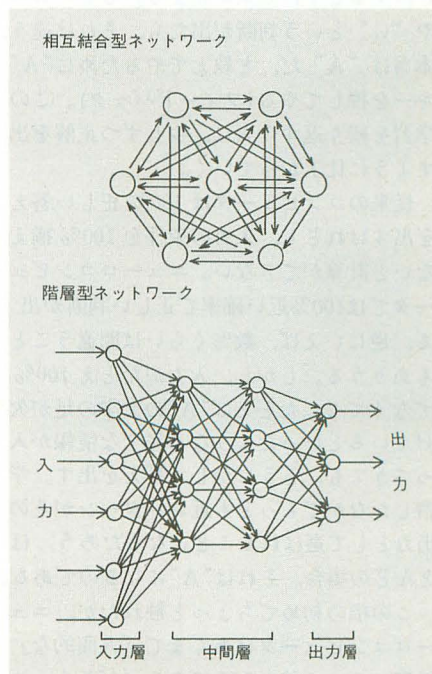



図4　フィードバックと学習

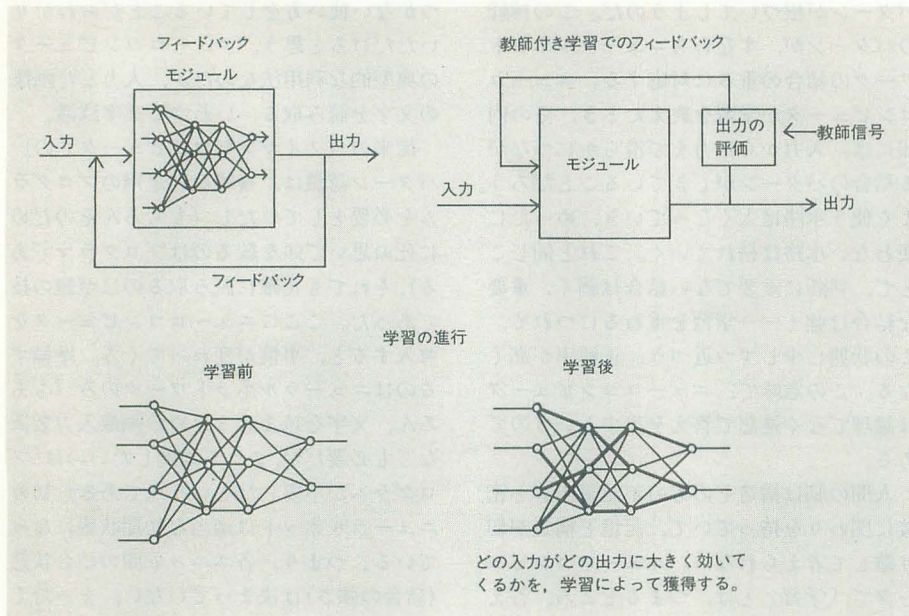
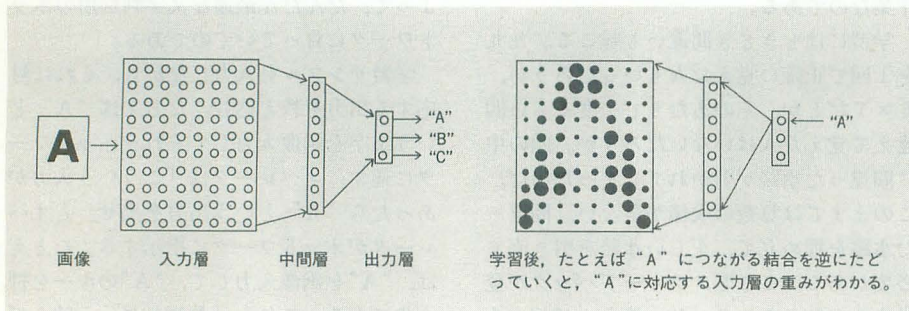



図5　文字認識

たとえば"A"という画像の入力に，システムが"A"と判断すれば，よくやった，偉い，というわけで，なにもキーを押さない。すると学習効果が高まる。もし"B"や"C"という判断が出たら，それは違う，本当は"A"だ，と教えてやるために"A"キーを押してやる(フィードバック)。この学習を繰り返すうちに，少しずつ正解を出すように仕上がっていく。

従来のコンピュータは100%正しい答えを出すけれども，入力の条件を100%揃えないと計算ができない。ニューロコンピュータでは100%近い確率で正しい判断が出る。逆にいえば，数%くらいは間違うこともありうる。しかし，入力がたとえ100%でなくても，たとえば"A"の文字の足が欠けているといったような不完全な情報が入ってきても，どうにかして答えを出す。学習したなかでもっとも近いパターンがその出力として選ばれることになるだろう。ほとんどの場合，それは"A"になるのである。

この項の初めでちょっと触れたが，ニューロコンピュータはあくまで「人間的な」処理において強力なのである。だから，ニューロコンピュータが現在のノイマン型コンピュータを駆逐することはあり得ないといい切っていい。むしろニューロコンピュータは，「数値的な」処理を得意とするノイマン型コンピュータと協力しあうべきだ。たとえば，スーパーコンピュータの端末の，マンマシンインタフェイスの部分にニューロコンピュータを置く。多少でたらめな操作をしてもニューロコンピュータはユーザーの意図するところを汲み取ろうとする。大型機はとかくユーザーフレンドリでないことが多いが，これで大幅改善だ。

パーソナルコンピュータにもニューロが組み込めるとうれしい。まずワードプロセッサ。日本語フロントプロセッサにはもっと利口になっていただきたいものだ。完成されたニューロコンピュータなら（いまのニューロコンピュータではまだ規模が足りない。人間の神経結合の数は，いまのニューロ素子の100万倍のオーダーなのだ）同音異義語もなんなくこなしてくれるに違いない。ゲームの分野でもいくらでも応用が効く。学習次第でいくらでも強くなる将棋の相手，本当の冒険をさせてくれるロールプレイングゲームのマスター，などなど。より高度な処理ができ，なおかつユーザーとの相性のよさを追求する方向に進むコンピュータ。ああバラ色の未来。

昨今のニューロブームを批判する向きもある。現在の程度のニューラルネットワークでは，いずれ限界がきてしまい，そこでブームは終わるという悲観的な予測もあるようだ。ニューロコンピュータそのものはかなり前から考えられていたのだが，数年前まで研究は下火だった。本格的な研究が始まって時間はほとんどたっていないことを考えると，性急に否定するのは早いと思う。確かにいまのままのモデルではなにかが足りないような気もするのだが，そこはそれ，数年後に出てくるであろうニューロコンピュータの実用機に期待するとしよう。評価はそれからでも遅くあるまい。

総括

"固い"コンピュータのスタイルと"柔らかい"コンピュータのスタイル，それぞれを見てきた。もちろん，こうした「未来志向」のコンピュータはまだまだ研究も始まったばかりだし，ノイマン型コンピュータもこれからますます発達させていかなくてはならないのはいうまでもない。

新しい研究が軌道に乗るまでは苦しいものだが，成果が上がらないからといってほんの何年かで下火になってしまうようでは，単なるブーム。これでは少し困る。

僕らのようにノイマン型コンピュータとつきあうことに身体が慣れてしまった者には（これも人間の脳の偉大な順応性のおかげ！），できるだけ性能がよく，なおかつ手頃なお値段のノイマン型コンピュータが出たほうがうれしい。

しかし，真の「電脳社会」を築くためには，非ノイマン型のアーキテクチャを持つコンピュータも望まれている。ここに述べたコンピュータは，その構造も動作原理も，ノイマン型に毒された（過激かな）脳味噌には新鮮な驚きを与えてくれるだろう。

◆参考文献

元岡達，渕川優，「第五世代コンピュータ」，岩波書店
村田健郎，小国力，唐木幸比古，「スーパーコンピュータ　科学技術計算への適用」，丸善
中野馨，飯沼一元，ニューロンネットグループ，桐谷滋，「入門と実習　ニューロコンピュータ」，技術評論社（ニューロコンピュータの技術的な面に詳しい。C言語で書かれたニューラルネットワークのシミュレーションプログラムも載っている）
日本経済新聞社編，「ニューロコンピュータ 90年代に実用機が登場」，日本経済新聞社

micro computer 入門

周辺LSIを使いこなそう(1)

Z80とその家族

CPUをパワーアップするために用意されたさまざまな周辺LSI。ここではZ80を支えているPIO，CTC，DMAを使用法をまじえて紹介しましょう。これらを使いこなすことで，CPUだけではできない高度な処理も可能になります。

Nishikawa Zenji
西川 善司

Z80CPUと周辺LSI

Z80は1976年に発表され，当時，主流となっていたCPU8080（インテル）とアッパーコンパチ，つまり，8080用に作られたプログラムがそのまま動き，しかも高速であるということから，一大センセーションを巻き起こしました。発表から10年以上経った今でも多くのパソコンのメインCPUとして使用され，これからもサブCPUとして，活躍していくことは間違いないといわれています。

さて，これほどまでにZ80の人気が衰えないのはZ80CPU自身の実力もさることながら，Z80CPUを支える，周辺LSIの性能も見逃せません。

ご存じの方も多くおられるとは思いますがZ80の割り込みモードには0～2の3つのモードがあります。その中でも，モード2は64Kバイトのメモリ空間の好きな場所に置かれたプログラムへ割り込み処理をさせることが可能です。

具体的にはIレジスタに割り込みベクトルの上位バイトをセットしておき，周辺LSIが割り込み要求をすると，Z80はそのLSIが先に記憶していた割り込みベクトル下位バイトとIレジスタをあわせてできるアドレスを参照し，そこへ処理を移します(詳しくは'89年2月号の特集の「割り込みってなんだろな」を読んでください)。しかも，割り込み中に割り込みがかかっても処理できるので8ビットながらかなり高度なプログラミングが可能です。Z80周辺LSIはすべてこの割り込みモード2に対応した設計がされているのです。

ではそれぞれのLSI個別に説明していきたいと思います。なお，ここで述べられたことがすべてではないので，興味のある方は参考文献を参照してください。

Z80 PIO

Z80PIO (PARALLEL I/O INTERFACE CONTROLLER) は最も初めに発表された，Z80用周辺LSIです。汎用性の高いLSIですので，すでに馴染みの深い方も多いことでしょう。PIOの仕事は簡単にいうと，CPUと周辺装置とのインタフェイスで，この周辺装置とは具体的にはジョイスティックやプリンタ，キーボードなどが代表的です。PIOが単なるインタフェイスと大きく異なるのは，やはりプログラム可能という点でしょう。

PIOはTTLコンパチブルの入出力ポートを2つ持っており，データのやりとりはこの2つのポートを使って行います。

このポートの使用法として，0～3の4つのモードがあり，それぞれ，

モード0	出力モード
モード1	入力モード
モード2	双方向モード
モード3	ビットモード

と呼ばれます。

これらのモードの設定は2ビットのモード制御用レジスタによって行います。図1にもあるように，下位4ビットをすべて1にしてPIOにデータを送ると，そのバイト中の第7，6ビットに書いてある内容がモード制御用レジスタの値として設定されます。

余った第4，5ビットは未使用で，1でも0でもかまいません。第7，6ビットの値によって，

D_7	D_6	モード
0	0	出力モード
0	1	入力モード
1	0	双方向モード
1	1	ビットモード

と設定されます。

2つのポートは，それぞれポートA，ポートBと呼ばれます。この2つのポートはほとんど完全に独立しており，割り込みベクトルもA，B別のものを設定できます。どうして，「ほとんど完全」といういい方をしたかというと，モード2はポートAにしか設定できません。理由はモード2の説明のところで行います。

●出力モード（モード0）

CPUから出力レジスタに対して書き込みが行われると，PIOは周辺装置へデータを出力したい，と要求します。すると周辺装置がデータを受け取り，「データ受け取り完了」といってきます。このとき，もし，割り込みスイッチがONになっているなら割り込みが発生します。

●入力モード（モード1）

まず，ポートに対して，無意味なリード動作を行ってやります。これが行われるとPIOは動作を開始し，周辺装置へデータがほしいと要求します。周辺装置はPIOへデータを送り，PIOはこれを受け取ります。このとき割り込みスイッチがONであれば割り込みが発生します。

●双方向モード（モード2）

ポートAの制御にポートBの端子を使用するため，ポートA専用のモードです。ポートAに対して，モード2が設定されると，ポートBはモード3でのみ，動作可能となります。

まず，ポートAに対してデータを書き込むと周辺装置へデータを渡したい，と要求し，周辺装置はデータを受け取ります。受

図1　Z80 PIO

D7	D6	D5	D4	D3	D2	D1	D0
ﾓｰﾄﾞ指定用ビット	ﾓｰﾄﾞ指定用ビット	×	×	1	1	1	1

モード制御レジスタへの設定
(注) ×は未使用ビット。よって0でも1でも構わない

D7	D6	D5	D4	D3	D2	D1	D0
割り込みスイッチ	AND／OR	HIGH LOW	ﾏｽｸ指定スイッチ	0	1	1	1

割り込み制御語の設定
D7＝1で割り込みON，D7＝0でオフ
D6＝1でAND　　，D6＝0でOR
D5＝1でHIGH　　，D5＝0でLOW
D4＝1でマスク指定　，D4＝0で指定しない
(注) D4～D6はモード3のときのみ使用

D7	D6	D5	D4	D3	D2	D1	D0
割り込みベクトル	割り込みベクトル	割り込みベクトル	割り込みベクトル	割り込みベクトル	割り込みベクトル	割り込みベクトル	0

割り込みベクトルの設定
(注) D0は必ず0、よって割り込みベクトルは必然的に偶数でなくてはいけない

D7	D6	D5	D4	D3	D2	D1	D0
割り込みスイッチ	×	×	×	0	0	1	1

割り込みスイッチの設定
(注) このコマンドによって、割り込みスイッチのみを設定できる

け取りが完了すると同時にポートAが割り込みONであれば割り込みが発生します。

次に，周辺装置がデータを転送したい，と要求し，PIOはこれを内部に受け取り，これ以上データは受け取れない，と周辺装置に知らせます。このとき，ポートBが割り込みオンであれば割り込みが発生します。

●ビットモード（モード3）

モード3は周辺装置の状態の変化を調べたりするのに適しており，MZ-2000などでは，キー割り込み処理にこのモードを使用しています。

モード3を設定したら，次に8ビットの入出力制御語を書き込まなければいけません。ビット=1で，そのビットは入力用ビットとして設定され，ビット=0でそのビットは出力用ビットとして設定されます。つまり，モード3設定後に，

$0F_H=00001111_B$

を書き込んだとすれば，上位4ビットは出力用，下位4ビットが入力用となります。ですから，たとえば上記のような例でモード3を動作させた場合，

$FF_H=11111111_B$

をPIOに書き込んだとしても，周辺装置には上位4ビットの1111_Bのデータしか送られないこととなります。また，読み込み動作のときには入力用と設定したビットデータと，最後に書き込んだ出力データの内容が返ってきます。

いままでの例を用いて考え，読み込み動作を行うと下位4ビットの読み込みデータと，上位4ビットの1111_Bが返ってくることになります。

モード3はモード0～2のときとは少し異なった割り込み制御語を設定しなければいけないので，次にこれについて説明します。割り込み制御語のD4ビット=1で次に8ビットのマスクビットを設定できます。マスクビットはビット=0でそのビットを状態検出用のビットとして設定します。ビット=1で，そのビットは検出用でないと設定され，その名のとおり，マスクされるわけです。

割り込み制御語のD5ビットはその状態検出ビットが1のときに割り込み発生条件を真とするのか，0のときに真とするのかを設定します。D6ビットは，設定された状態検出用ビットがすべてD5ビットで設定したビットと同じでなければ割り込み条件が真とならない（AND）のか，どれかひとつのビットでもD5ビットで設定したビットと同じであれば割り込み条件が真となるのか（OR）を設定します。

このモードはCPUが周辺装置の見張り役に徹しなくてもすむため効率のよいプログラム設計が可能になります。

＊　＊　＊

以上でPIOの解説を終わります。文中にいくつか「PIOへ～を要求する」とありましたが，これらは実際はもっと厳密なパルスのやりとりがあります。ここでは，観念的に理解してほしいので，そういう説明は省きました。ご了承ください。

Z80 CTC

CTC（COUNTER TIMER CIRCUIT）はX1シリーズのFM音源ボードなどに装着されており，そのため，X1ユーザーにはすでにお馴染みのLSIですね。Z80CTCは0～3の4つのチャンネルを持ち，それらは独立したカウンタ，割り込みベクトルを持っています。

一定のリズムで割り込みを起こせるので，Z80があたかも同時にいくつものプログラムを実行しているかのごとく動作させることも可能です。

図2のCTCの制御語を理解してしまえばCTCを理解したのも同然ですので，以下はその制御語について話を進めていきます。

制御語は8ビットで構成されており，それぞれのビットが0または1で意味が変わってきます。

●D0ビット

このビットは通常1でなければなりません。チャンネル0に対して，ここを0で制御語を送ると，これは割り込みベクトルであるとCTCは判断します。チャンネル1～3の割り込みベクトルもこのとき自動的に決まります。すなわち，

チャンネル0は　$*****000_B$
チャンネル1は　$*****010_B$
チャンネル2は　$*****100_B$
チャンネル3は　$*****110_B$

となるわけです。

●D1ビット

このビット=1でCTCは動作を停止します。ですから，

$00000011_B=03_H$

をCTCのリセットコマンドとして，使うことが多いようです。

●D2ビット

このビット=1で次に書き込まれた8ビットデータがタイムコンスタント（カウンタ）であることを表します。

●D3ビット

このビット=1でCTCはトリガによって動作開始します。このビット=0で，タイムコンスタントが書き込まれた直後動作開始します。

また，このビットは後述するタイマモードでのみ意味を持ちます。

●D4ビット

このビットによって，トリガの極性を決めます。ビット=1でパルスの立ち上がりをトリガとみなし，ビット=0でパルスの立ち下がりをトリガとみなします。後述のカウンタモードではここで設定されたトリガくるたびにカウンタをカウントダウンします。ジョイスティックのボタンをトリガと呼んだりしますがここでいうトリガも似たようなものです。パルスの変化をボタンが押されたり，ボタンから指が離れたりしたものと考えればわかりやすいでしょう。

●D5ビット

タイマモードに設定したときのみ意味をなすビットです。ダウンカウンタの値をビット=0で16倍，ビット=1で256倍します。

●D6ビット

ここで，カウンタモード（ビット=1），タイマモード（ビット=0）を決めます。

カウンタモード：

このモードは先程のトリガによってカウンタをカウントダウンするモードです。

タイマモード：

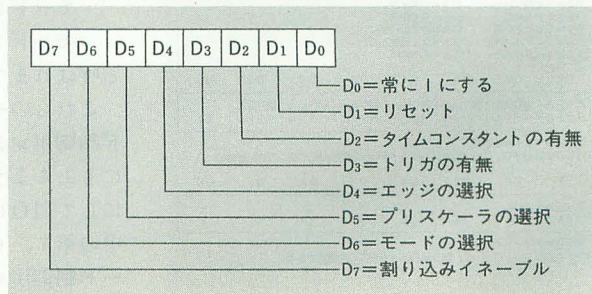
システムクロックをトリガとして，カウンタをカウントダウンするモードです。

●D7ビット

割り込みのスイッチビットです。ビット=1でカウンタ=0で割り込みが発生します。ビット=0にすると割り込みは発生しません。

D2=1の制御語の後ろに書き込むタイムコンスタント（カウンタ）は1が最小です。0は256とみなされるので注意が必要です。

ではCTCのサンプルです。画面に星を流すというプログラムです。CZ-8FB01でCLEAR &HD000を実行後，このプログラム

図2　CTCのチャンネルコントロールワード

をロードしてCALL &HD000を実行してください（画面が出ないときはctrl-D)。前にも似たようなことをやりましたが今度はグラフィック版です。

Z80 DMA

CPUが単なるデータ転送のために仕事を頻繁に中断していたのでは効率がよくありません。そんな問題を解決するために生まれたのがDMA(DIRECT MEMORY ACCESS)で，実にデータ転送における地位はCPUをも凌ぐというLSIです。また，データの転送と組み合わせて，特定もしくは不特定データのサーチも可能なためかなり複雑なことができます（たとえばあるデータを見つけるまでデータを読み出せとか)。

さて，具体的な速さですが，4MHzのクロックでDMAを動かすとすれば1秒間に2Mバイトのデータを転送可能です。これはZ80CPUがデータ転送を行う速さの10倍以上ということになります。

図4を見ながらライトレジスタ個別に説明していきます。

●ライトレジスタ0

DMAの仕事の内容を具体的に決定してやるのが第0，1ビットです。選べる仕事の内容というのは転送，サーチ，転送&サーチです。

DMAには2つのポートA，Bがあり，それぞれをソース（転送元）にするのかデスティネーション（転送先）にするのかを決めてやるのが第2ビットです。

第3，4ビットはポートAのアドレスを更新するのかしないのかを決定してやります。上位バイトのみ更新したり（第4ビット＝1)，下位バイトのみ更新したり（第3ビット＝1）することも可能です。

第5，6ビットは処理バイト数（転送したりサーチしたりするバイト数）を更新するかしないかを決定してやります。この処理バイト数指定も上位バイト，下位バイトに分けて設定可能です。

この更新スイッチはほかのライトレジスタにも使用されていますがいずれも＝1で更新する，＝0で更新しない，です。

更新する値があるならば，このライトレジスタ設定後その値を送ってやる必要があります。たとえば第3～6ビットすべて＝1ならばポートAアドレス下位，ポートAアドレス上位，処理バイト数下位，処理バイト数上位の順にDMAに送ってやります。もし更新しないものがあるならば上の例から更新しないものを省いてDMAに送ってやればいいことになります。

ここで注意点がひとつあります。処理バイト数は希望する処理バイト数－1をセットしてやらなければいけません。たとえば処理バイト数を1バイトにしたい場合，1－1＝0で0000ʜをDMAに送ればいいことになります。

●ライトレジスタ1

ポートAはメモリとするのか，I/Oとするのか，ポートAのアドレス値は変化させるのか，また，どう変化させるのかを決定してやります。

●ライトレジスタ2

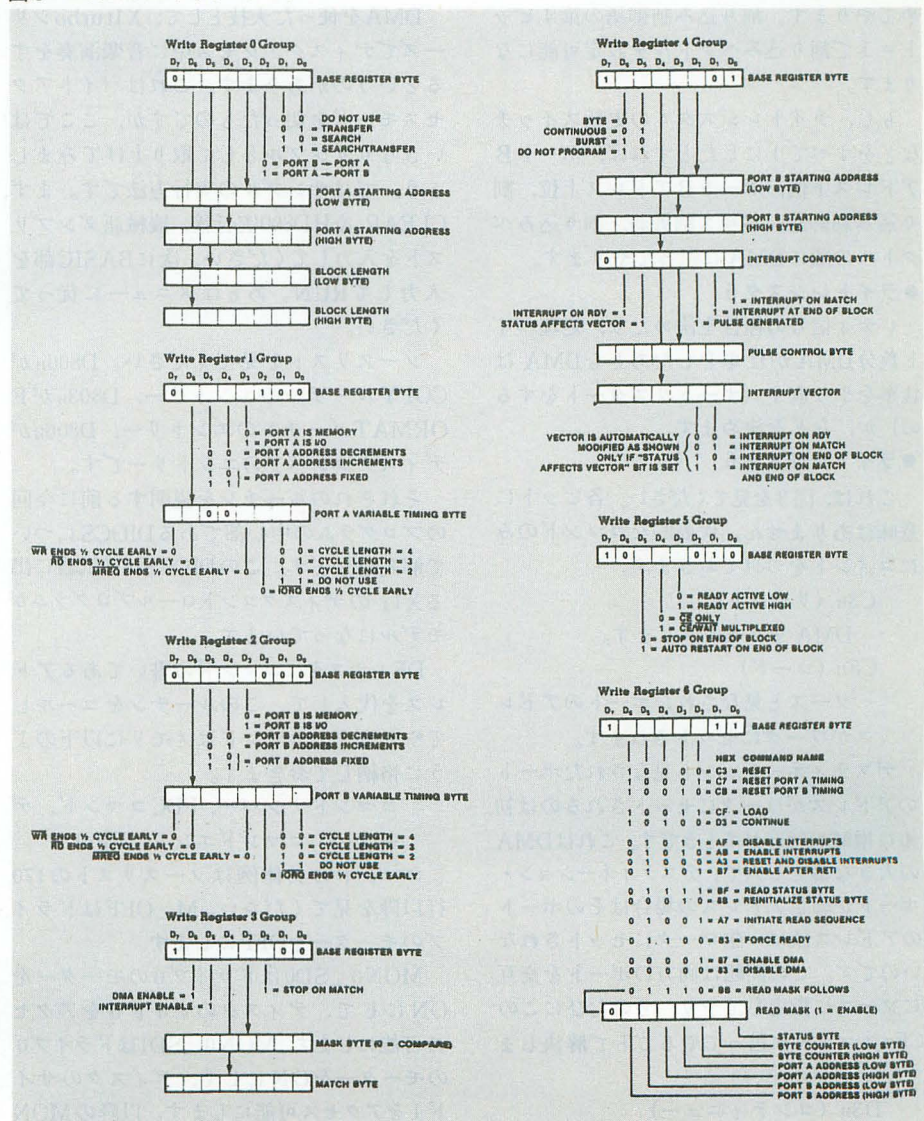
ポートBのライトレジスタ1と考えて結構です。

●ライトレジスタ3

主にサーチのときに使われるライトレジスタです。第6ビットはDMAを動作開始にするかどうかのスイッチです。第5ビットは割り込みスイッチ，第4ビットはサーチ対象にするバイトデータを更新するかどうかのスイッチです。第3ビットはマスクバイトを更新するかどうかのスイッチで，第2ビットでDMAがサーチ対象バイトを発見したときDMAはどうしたらよいのかを決めてやります。＝1でサーチ対象バイトを発見したらDMAを停止します。＝0ではDMAは動作を続けます。

更新するバイトがあるときはこのライトレジスタ設定後に送ってやる必要があります。順番はマスクバイト，サーチ対象バイトです。

さて，マスクバイトというのが出てきましたが，これについて少し解説しておきます。マスクバイトとは不特定データをサーチするときに使うものでバイト中のビット＝1でそのビットはマスクされます。つまり，マスクしない（特定データをサーチしたい）ときはマスクバイト＝00ʜにすればいいことになります。具体例をひとつ挙げる

図3 Z80DMAのライトレジスタ

と，マスクバイト＝11110000$_B$でサーチ対象バイト＝10100000$_B$にしたとすると，DMAは00$_H$をサーチします。

●ライトレジスタ4

第2，3ビットはポートBのアドレスの更新スイッチです。第5，6ビットでDMAの動作モードを決めます。バイトモードは1バイト転送するたびにCPUにバスが返ってくる（CPUが動けるようになる）モード。バーストモードはレディ信号が有効のあいだはDMAが動作を続ける（つまりレディ信号が無効になるとDMAは停止する）モード，コンティニュアスモードは処理バイト数分DMAが仕事を終わるまで，CPUが気絶している（？）モードです。速さはコンティニュアスモード>バーストモード>バイトモードとなります。

第4ビットは割り込み制御語の更新スイッチです。DMAもステータス・アフェクツ・ベクトル・モードを持っていますのでこの制御語でこのモードにするのかどうかを決めてやります。割り込み制御語の第4ビット＝1で割り込みベクトルを設定可能になります。

もし，ライトレジスタ4の更新スイッチなどをすべて1にしたとすれば，ポートBアドレス下位，ポートBアドレス上位，割り込み制御語，パルス制御語，割り込みベクトルの順にDMAに送ってやります。

●ライトレジスタ5

レディ信号の極性を決めたり，処理バイト数分DMAが仕事をしたあともDMAは仕事をやり直す（オートリスタートをするの）か，などを決めます。

●ライトレジスタ6

これは，図3を見てください。各ビットに意味はありません。代表的なコマンドのみにコメントをつけておきます。

C3$_H$（リセット）

DMAをリセットします。

CF$_H$（ロード）

ソースと見なされたポートのアドレスがワークにセットされます。

デスティネーションと見なされたポートのアドレスがワークにセットされるのは初めに増減が行われるときです。これはDMAの大きな落とし穴で，デスティネーション・ポートが固定アドレスの場合はそのポートのアドレス値は一生ワークにセットされないのです。この問題は両方のポートを交互にソースに指定してやり，そのたびにこのCF$_H$コマンドを送ってやることで解決します。

D3$_H$（コンティニュー）

サーチなどで一度動作終了したDMAを再び動作開始させるコマンドです。

AF$_H$（DI）

DMAの割り込みを禁止します。

AB$_H$（EI）

DMAの割り込みを許可します。

B3$_H$（フォースレディ）

DMAをレディ信号と関係なく動作させます。

83$_H$（ディスイネーブル）

DMAを停止させます。

87$_H$（イネーブル）

DMAを動作ONにします。ライトレジスタやリードレジスタに書き込みが行われるとDMAは停止してしまうのでこのコマンドは最後に送らなければいけません。

*　　*　　*

なお，リードレジスタの説明は紙面の都合により省かせていただきます。

*　　*　　*

DMAを使った大技として，X1turboシリーズでディスクアクセス中に音楽演奏をするというのがあります。これはバイトアクセスモードを使ったものですが，ここではいきなりサンプルとして取り上げてみましょう。ではサンプルの実行方法です。まず，CLEAR &HD800実行後，機械語ダンプリストを入力してください。次にBASIC部を入力してRUN。あとはメニューに従ってください。

ソースリストを見てください。D800$_H$がCOPYルーチンのエントリー，D803$_H$がFORMATルーチンのエントリー，D806$_H$がディスクコンペアのエントリーです。

それぞれのルーチンを説明する前に今回のプログラムの中心部であるDIOCSについて解説をします。このDIOCSは『試験に出るX1』のディスクコントロールプログラムがモデルになっています。

DEレジスタにコマンドの書いてあるアドレスを代入して，このルーチンをコールしてやります。コマンドはメモリに以下のように格納しておきます。

コマンドナンバー，FDCコマンド，データ……コマンドエンド（FF$_H$）

コマンドの具体例はソースリストの170行以降を見てください。M_OFFはドライブのモーターをOFFにします。

MON0_SD0はドライブ0のモーターをONにして，ディスクのサイド0をアクセス可能にします。MON0_SD1はドライブ0のモーターをONにして，ディスクのサイド1をアクセス可能にします。以降のMON1_SD0，MON1_SD1も同じです。R_COMは1トラック分（1000$_H$バイト）データを読み込みます。W_COMは1トラック分データを書き込みます。RESTORE0，1はドライブのリストアを行います。FDCのコマンドの意味そのものはここでは省略します。

DMAのコマンド列はソースリスト820行付近にあります。X1turboのDMAのレディ信号はFDCとつながっていますので，DMAはバイトモードで動かしています。

例として，R_DMAのコマンド列を解説します。これはDMAをディスクからデータを読み出すようにプログラムしてやっている例です。後ろにある書き込み用のDMAコマンド列W_DMAもほとんど同じです。

83$_H$はDMAストップ，7D$_H$はWR0の設定をしています。7D$_H$＝01111101$_B$です。よって転送モード，方向はポートA→ポートB，ポートAのアドレス，処理バイト数を更新する，です。よって，この後ろにポートAアドレス，処理バイト数のワード値がきています（FB$_H$，0F$_H$，FF$_H$，00$_H$）。0FFB$_H$はFDCのデータレジスタのI/Oアドレスです。00FF$_H$は1セクタのバイト数−1です。

2C$_H$は，WR1でポートAは固定でI/Oと指定しています。10$_H$はWR2でポートBはアドレスをインクリメントモードにして，メモリに割り当てています。80$_H$はWR3ですがなにも設定していません。

8D$_H$はWR4でDMAの動作モードをバイトモードにして，ポートBのアドレスを更新する，としています。よってこの後ろにポートBのアドレス値（ワード）がきています。実際に読んだデータをメモリのどこに格納するのかを決めてやっているのがこのポートBのアドレス値です。

92$_H$はWR5です。レディ信号の極性を決めています。X1turboではFDCがデータをリクエストするとLowのパルスがくるのでレディ信号の極性はLowにしてやります。CF$_H$はWR6はロードコマンド，87$_H$もWR6のコマンドでDMA動作開始を意味します。

さて，それぞれのルーチンでやっていることは大したことはなく（ほとんどDIOCSをCALLしているだけ）ドライブ0のディスクから2トラック分のデータをメモリに読み込んで，ドライブ1へそのデータ書き込んでいるだけです。

コンペアは2枚のディスクを交互に読んでデータのXOR値が一致しているかをチェックしてデータが同一のものかを判定しているだけです。

FORMATはDIOCSを一切コールせず，

独立しています。初めにMAKE_Fというところでフォーマットデータを作成し，あとはフォーマットデータにIDをセットしながらそのフォーマットデータを順番にライトトラックしてやっているだけです。DMAコマンド列も独立して持っていますが，実は処理バイト数が28B0Hになっただけで，W_DMAと同じです。

プログラムの特徴として，DMAを使っているためMusicBASICなどでBGMを鳴らしながらディスクをフォーマットしたりコピーしたりできます。また，'89年2月号のサンプル「星が流れる」などを実行しながら本プログラムが実行可能です。

これを機会に皆さんもZ80の周辺LSIを使いこなして，MZ/X1シリーズを盛り上げていきましょう。

◆参考文献

額田忠之，「Z80ファミリ・ハンドブック」CQ出版社
祝一平，「試験に出るX1」，日本ソフトバンク
「MZ-2000オーナーズマニュアル」，シャープ

リスト1

```
D000 CD 24 D0 21 53 D1 22 5A : 82
D008 00 F3 ED 4B 20 D0 3E 58 : B1
D010 ED 79 ED 4B 22 D0 3E A7 : 75
D018 ED 79 3E 80 ED 79 FB C9 : 4E
D020 A0 1F A1 1F DD 21 C4 D2 : 13
D028 06 28 C5 CD 37 D0 11 0F : E7
D030 00 DD 19 C1 10 F4 C9 CD : 51
D038 49 D2 7D B7 EA DC D0 11 : F6
D040 33 00 CD 87 D2 7B D6 80 : 2A
D048 7A DE 02 30 04 3E F8 18 : DC
D050 0A EB 11 80 02 B7 ED 52 : 7E
D058 EB 3E 08 32 A3 D2 7B D6 : 29
D060 40 7A DE 01 30 3B 21 40 : 65
D068 01 B7 ED 52 E5 11 00 32 : 1F
D070 EB CD 87 D2 DD 73 02 DD : 40
D078 36 06 FF 3A A3 D2 DD 77 : 3E
-----------------------------------
SUM: 9A 0A 1D 63 A0 7E 3D 67 E652

D080 07 E1 7D D6 64 7C DE 00 : F9
D088 30 0C DD 36 04 00 3A A3 : 30
D090 D2 DD 77 05 18 08 DD 36 : 5E
D098 04 FF DD 36 05 00 C3 21 : FF
D0A0 D1 21 80 02 B7 ED 52 E5 : 4F
D0A8 11 00 32 EB CD 87 D2 DD : 31
D0B0 73 02 DD 36 06 01 3A A3 : 6C
D0B8 D2 DD 77 07 E1 7D D6 64 : C5
D0C0 7C DE 00 30 0C DD 36 04 : AD
D0C8 00 3A A3 D2 DD 77 05 18 : 20
D0D0 08 DD 36 04 01 DD 36 05 : 38
D0D8 00 C3 21 D1 11 A4 00 CD : 37
D0E0 87 D2 7B D6 C8 7A DE 00 : CA
D0E8 30 04 3E FF 18 0A EB 11 : 8F
D0F0 C8 00 B7 ED 52 EB 3E 01 : E8
D0F8 32 A3 D2 7B FE 64 38 04 : C0
-----------------------------------
SUM: 69 FA F0 85 1B 1E 9C C7 7632

D100 3E 08 18 02 3E F8 F5 21 : AC
D108 00 A0 CD 87 D2 DD 73 02 : 18
D110 3A A3 D2 DD 77 06 DD 77 : 5D
D118 04 F1 DD 77 07 DD 36 05 : 68
D120 00 DD 36 00 E8 DD 36 01 : 0F
D128 E3 DD 36 03 00 DD 36 0A : 16
D130 04 DD 36 0B 03 CD 49 D2 : 0D
D138 7C E6 07 3C DD 77 09 DD : DF
D140 86 0A DD 77 08 3E 80 DD : 87
D148 77 0C DD 36 0D 28 DD 36 : DE
D150 0E 0C C9 F3 ED 73 0E D2 : 16
D158 31 C4 D2 F5 C5 D5 E5 DD : 18
D160 E5 3E 28 DD 21 C4 D2 F5 : D4
D168 DD 7E 08 3D DD 77 08 C2 : BE
D170 FD D1 DD 4E 00 DD 46 01 : 1D
D178 AF ED 79 DD 7E 0B 3D DD : 95
-----------------------------------
SUM: 89 19 18 01 99 87 E6 B0 6C86

D180 77 0B 20 0E 3E 03 DD 77 : 45
D188 0B DD 7E 0A B7 28 03 DD : 2F
D190 35 0A DD 7E 09 DD 86 0A : 10
D198 DD 77 08 DD 7E 03 DD 86 : 1D
D1A0 02 DD 77 03 FA AF D1 DD : B0
D1A8 5E 05 DD 56 04 18 0A DD : 99
D1B0 36 03 00 DD 5E 07 DD 56 : AE
D1B8 06 78 83 B7 F2 13 D2 FE : 8D
D1C0 C0 DA 2C D2 47 7A B7 FA : 0A
D1C8 DF D1 DD CB 0C 0E 30 22 : C4
D1D0 03 DD 7E 0D 3C FE 50 D2 : C7
D1D8 43 D2 DD 77 0D 18 13 DD : 7E
D1E0 CB 0C 06 30 0D 0B DD 7E : 80
D1E8 0D 3D FE FF CA 43 D2 DD : 03
D1F0 77 0D DD 7E 0C ED 79 DD : 2E
D1F8 71 00 DD 70 01 11 0F 00 : DF
-----------------------------------
SUM: D5 76 7C 9E 4A D6 4E F5 738E

D200 DD 19 F1 3D C2 67 D1 DD : FB
D208 E1 E1 D1 C1 F1 31 00 00 : 76
D210 FB ED 4D DD 7E 0E 3C FE : D8
D218 19 D2 43 D2 DD 77 0E 60 : C2
D220 69 01 B0 37 B7 ED 42 44 : 7B
D228 4D C3 C5 D1 DD 7E 0E 3D : 4C
D230 FE FF CA 43 D2 DD 77 0E : 3E
D238 60 69 01 B0 37 09 44 4D : 4B
D240 C3 C5 D1 CD 37 D0 C3 02 : F2
D248 D2 ED 5B 5D D2 ED 4B 60 : E1
D250 D2 CD 62 D2 22 5D D2 ED : 11
D258 5F 32 5F D2 C9 23 E1 00 : 8F
D260 E7 03 21 00 00 3E 10 29 : 82
D268 CB 23 CB 12 D2 70 D2 09 : E8
D270 3D C2 67 D2 C9 3E 10 44 : 93
D278 4D 21 00 00 29 EB 29 EB : 96
-----------------------------------
SUM: E8 9F D2 5A 63 82 02 C7 328A

D280 30 01 09 3D 20 F6 C9 3E : 94
D288 10 42 4B EB 21 00 00 29 : D2
D290 EB 29 EB 30 01 23 B7 ED : F7
D298 42 30 03 09 18 01 13 3D : E7
D2A0 20 ED C9 00 00 00 00 00 : D6
D2A8 00 00 00 00 00 00 00 00 : 00
D2B0 00 00 00 00 00 00 00 00 : 00
D2B8 00 00 00 00 00 00 00 00 : 00
D2C0 00 00 00 00             : 00
-----------------------------------
SUM: 8D 89 0B 61 5A 1A 93 91 6F3E
```

リスト2

```
D800 C3 0A D8 C3 C4 D9 C3 94 : 5C
D808 D8 00 F3 ED 73 E8 DD 31 : 21
D810 6A DE FB AF 32 B2 D9 11 : C0
D818 B5 D9 CD E1 DB 11 BC D9 : BD
D820 CD E1 DB 01 00 28 C5 C5 : 3C
D828 11 42 D9 CD E1 DB C1 AF : 25
D830 32 C3 D9 3A B2 D9 32 B3 : 78
D838 D9 79 32 B2 D9 CD C0 DB : 77
D840 11 B0 D9 CD E1 DB 21 00 : 44
D848 DF 22 C2 DD 11 4E D9 CD : A5
D850 E1 DB 11 45 D9 CD E1 DB : 74
D858 11 4E D9 CD E1 DB 11 48 : 1A
D860 D9 CD E1 DB 3E 01 32 C3 : 96
D868 D9 11 B0 D9 CD E1 DB 21 : 1D
D870 00 DF 22 C2 DD 11 7F D9 : 09
D878 CD E1 DB 11 4B D9 CD E1 : 6C
-----------------------------------
SUM: 04 B9 65 3D 8F CA F2 3F 3D6E

D880 DB 11 7F D9 CD E1 DB C1 : 8E
D888 0C 10 9B 11 3D D9 CD E1 : 8C
D890 DB C3 67 DD F3 ED 73 E8 : 1D
D898 DD 31 6A DE FB AF 32 B2 : E4
D8A0 D9 11 B5 D9 CD E1 DB 11 : 12
D8A8 BC D9 CD E1 DB 01 00 28 : 47
D8B0 C5 C5 11 42 D9 CD E1 DB : 3F
D8B8 C1 AF 32 C3 D9 3A B2 D9 : 03
D8C0 32 B3 D9 79 32 B2 D9 CD : C1
D8C8 C0 DB 11 B0 D9 CD E1 DB : BE
D8D0 21 00 DF 22 C2 DD 11 4E : 20
D8D8 D9 CD E1 DB 11 45 D9 CD : 5E
D8E0 E1 DB 11 4E D9 CD E1 DB : 7D
D8E8 CD 2A D9 F5 11 48 D9 CD : C4
D8F0 E1 DB 3E 01 32 C3 D9 11 : DA
D8F8 B0 D9 CD E1 DB 21 00 DF : 12
-----------------------------------
SUM: E5 87 4F AF 27 D9 F2 84 FF84

D900 22 C2 DD 11 4E D9 CD E1 : A7
D908 DB 11 4B D9 CD E1 DB 11 : AA
D910 4E D9 CD E1 DB CD 2A D9 : 80
D918 E1 BC C2 30 DD C1 0C 10 : 49
D920 8F 11 3D D9 CD E1 DB C3 : 02
D928 67 DD 21 00 DF 11 00 20 : 75
D930 0E 00 79 AE 23 1B 4F 7A : 3C
D938 B3 20 F7 79 C9 04 00 04 : 14
D940 01 FF 04 80 FF 04 90 FF : 16
D948 04 81 FF 04 91 FF 02 80 : 9A
D950 01 02 80 02 02 80 03 02 : 0C
D958 80 04 02 80 05 02 80 06 : 93
D960 02 80 07 02 80 08 02 80 : 95
D968 09 02 80 0A 02 80 0B 02 : 24
D970 80 0C 02 80 0D 02 80 0E : AB
D978 02 80 0F 02 80 10 FF 03 : 25
-----------------------------------
SUM: F6 0A A2 8F 11 78 A9 56 3907

D980 A0 01 03 A0 02 03 A0 03 : EC
D988 03 A0 04 03 A0 05 03 A0 : F2
D990 06 03 A0 07 03 A0 08 03 : 5E
D998 A0 09 03 A0 0A 03 A0 0B : 04
D9A0 03 A0 0C 03 A0 0D 03 A0 : 02
D9A8 0E 03 A0 0F 03 A0 10 FF : 72
D9B0 01 1C 00 00 FF 04 80 00 : A0
D9B8 00 04 00 FF 04 81 00 00 : 88
D9C0 04 01 FF 00 F3 ED 73 E8 : 3F
D9C8 DD 31 6A DE FB 21 00 DF : 51
D9D0 22 C2 DD 22 AC DB 22 B1 : 3D
D9D8 DB CD B1 DA 01 FF 0F ED : 2F
D9E0 78 11 BC D9 CD E1 DB AF : 56
D9E8 32 AA DB 32 AB DB CD 4C : 88
D9F0 DB 3A AA DB CD C0 DB 32 : 34
D9F8 AB DB AF 32 A9 DB 01 FC : E8
-----------------------------------
SUM: 69 01 3D 4D DE 1C 06 DE 250D

DA00 0F 3A A9 DB FE 01 20 04 : F0
DA08 1E 10 18 02 1E 00 3E 81 : 25
DA10 B3 ED 79 CD CC DC CD 14 : 6F
DA18 DB 16 11 21 AF DB CD F7 : 71
DA20 DC 01 F8 0F 3E F0 ED 79 : 78
DA28 CD CC DC 16 01 21 AF DB : 37
DA30 CD F7 DC 3A A9 DB 3C 32 : CC
DA38 A9 DB FE 01 28 C0 3A AA : 4F
DA40 DB 3C 32 AA DB FE 28 20 : 14
DA48 A5 01 FC 0F 3E 01 ED 79 : 56
DA50 11 01 DF 62 6B 2B 36 FF : 1E
DA58 01 FF 1F ED B0 11 01 ED : BB
DA60 62 6B 2B 36 00 01 FF 01 : 2F
DA68 ED B0 21 00 ED 36 01 23 : 05
DA70 36 8F 11 51 ED 62 6B 2B : 0C
DA78 36 8F 01 2F 00 ED B0 11 : A3
-----------------------------------
SUM: 27 62 83 E9 B5 25 71 A5 A3CF

DA80 BC D9 CD E1 DB 11 48 D9 : 50
DA88 CD E1 DB 3E 01 32 C3 D9 : 96
DA90 21 00 DF 22 C2 DD 11 7F : 51
DA98 D9 CD E1 DB 11 4B D9 CD : 64
DAA0 E1 DB 11 7F D9 CD E1 DB : AE
DAA8 11 3D D9 CD E1 DB C3 67 : DA
DAB0 DD DD 21 00 FE DD 36 00 : EC
DAB8 00 21 67 DB CD D2 DA 06 : E2
DAC0 10 C5 21 6C DB CD D2 DA : B6
DAC8 C1 10 F6 21 92 DB CD D2 : F4
DAD0 DA C9 22 A7 DB 2A A7 DB : F3
DAD8 4E 23 46 23 78 B1 C8 7E : 49
DAE0 23 22 A7 DB FE FF 28 03 : EF
DAE8 57 18 1A DD 34 00 16 00 : B0
DAF0 DD 5E 00 CB 23 DD E5 DD : C8
DAF8 19 ED 5B AC DB DD 73 00 : 38
-----------------------------------
SUM: BB E3 75 C9 24 FE 4D 2B 72EE

DB00 DD 72 01 DD E1 2A AC DB : BF
DB08 72 23 0B 78 B1 20 F9 22 : 04
DB10 AC DB 18 C1 3E 01 32 AE : 7F
DB18 DB DD 4E 00 DD E5 3A AE : B0
DB20 DB 16 00 5F CB 23 DD 19 : 34
DB28 DD 6E 00 DD 66 01 DD E1 : 4D
DB30 3A AA DB 77 23 3A A9 DB : 17
DB38 77 23 3A AE DB 77 23 36 : 2D
DB40 01 3A AE DB B9 D0 3C 32 : BB
DB48 AE DB 18 D0 01 FB 0F 3A : B6
DB50 AA DB ED 79 01 F9 0F 3A : 2E
DB58 AB DB ED 79 01 F8 0F 3E : 32
DB60 1C ED 79 CD CC DC C9 20 : E0
DB68 00 4E 00 00 0C 00 00 03 : 5D
DB70 00 F5 01 00 FE 04 00 FF : F7
DB78 01 00 F7 16 00 4E 0C 00 : 68
-----------------------------------
SUM: 60 99 98 F7 6E EF D5 6A 6295

DB80 00 03 00 F5 01 00 FB 00 : F4
DB88 01 E5 01 00 F7 36 00 4E : 62
DB90 00 00 0A 01 4E 00 00 01 : 5A
DB98 02 03 04 05 06 07 08 09 : 2C
DBA0 0A 0B 0C 0D 0E 0F 10 00 : 5B
DBA8 00 00 00 00 00 00 00 83 : 83
DBB0 79 00 00 AF 28 14 28 80 : 0C
DBB8 8D FB 0F 92 CF 05 CF 87 : 53
DBC0 F5 C5 E5 87 21 40 36 06 : C3
DBC8 00 4F 09 4D 44 3E 87 ED : 9B
DBD0 79 03 ED 79 3E 87 CB A0 : 12
DBD8 ED 79 0B ED 79 E1 C1 F1 : 6A
DBE0 C9 ED 53 C6 DD 1A FE FF : C3
DBE8 C8 13 ED 53 C4 DD 21 08 : E5
DBF0 DC 16 00 87 5F 19 5E 23 : 72
DBF8 56 EB ED 5B C4 DD CD 07 : FE
-----------------------------------
SUM: 31 82 3D 7E 31 38 9D 97 578B

DC00 DC ED 5B C4 DD 18 DE E9 : A4
DC08 12 DC 21 DC 4B DC 83 DC : 71
DC10 BB DC 1A 13 ED 53 C4 DD : A5
DC18 01 F8 0F ED 79 CD CC DC : E3
DC20 C9 EB 7E 23 56 23 5E 23 : 4F
DC28 22 C4 DD 01 F9 0F ED 59 : 12
DC30 01 FB 0F ED 51 01 F8 0F : 51
DC38 ED 79 CD CC DC 01 F8 0F : E3
DC40 ED 78 32 09 D8 E6 18 C2 : 38
DC48 24 DD C9 D9 2A C2 DD 22 : 8E
```

```
DC50 D2 DD 01 00 01 09 22 C2 : 9E
DC58 DD 21 C8 DD 16 0F CD F7 : 8C
DC60 DC D9 CD E4 DC 01 F8 0F : 4A
DC68 ED 79 CD CC DC 21 C8 DD : A1
DC70 16 01 CD F7 DC 01 F8 0F : BF
DC78 ED 78 32 09 D8 E6 1C C8 : 42
----------------------------------
SUM: 0F DE 39 EC 8F 11 E4 78 D091

DC80 C3 01 DD D9 2A C2 DD 22 : 65
DC88 D9 DD 01 00 01 09 22 C2 : A5
DC90 DD 21 D7 DD 16 11 CD F7 : 9D
DC98 DC D9 CD E4 DC 01 F8 0F : 4A
DCA0 ED 79 CD CC DC 21 D7 DD : B0
DCA8 16 01 CD F7 DC 01 F8 0F : BF
DCB0 ED 78 32 09 D8 E6 3C C8 : 62
DCB8 C3 18 DD 1A 13 ED 53 C4 : E9
DCC0 DD 01 FC 0F ED 79 B7 F0 : F6
DCC8 CD CC DC C9 C5 06 20 10 : 39
DCD0 FE 01 F8 0F ED 78 4F E6 : A0
DCD8 81 20 F6 79 C1 C9 3E 07 : DF
DCE0 3D 20 FD C9 EB 7E 23 56 : 05
DCE8 23 22 C4 DD 01 FA 0F ED : DD
DCF0 51 F5 CD CC DC F1 C9 01 : 76
DCF8 80 1F 04 ED A3 15 20 FA : 62
----------------------------------
SUM: 62 26 83 3F 8B 10 A1 8D 9886

DD00 C9 16 95 3A C3 D9 B7 20 : 21
DD08 05 21 72 DD 18 03 21 87 : 38
DD10 DD CD 3C DD 3E FF 18 3E : 56
DD18 16 95 21 9C DD CD 3C DD : 2B
DD20 3E FF 18 32 16 95 21 AA : FD
DD28 DD CD 3C DD 3E FF 18 26 : 3E
DD30 16 95 21 B7 DD CD 3C DD : 46
DD38 3E FF 18 1A 4E 23 46 23 : 49
DD40 7E B7 C8 ED 79 03 ED 79 : CC
DD48 CB A0 ED 51 0B ED 51 CB : BD
DD50 E0 03 03 23 18 EA 32 09 : 46
DD58 D8 11 B5 D9 CD E1 DB 11 : 11
DD60 BC D9 CD E1 DB 18 04 AF : E9
DD68 32 09 D8 F3 ED 7B E8 DD : 33
DD70 FB C9 F6 36 52 45 41 44 : 0C
DD78 20 45 52 52 4F 52 20 44 : 0E
----------------------------------
SUM: 3A 54 4B 06 47 11 7F 04 C00E

DD80 52 49 56 45 20 30 00 F6 : 7C
DD88 36 52 45 41 44 20 45 52 : 09
DD90 52 4F 52 20 44 52 49 56 : 48
DD98 45 20 31 00 02 37 57 52 : 78
DDA0 49 54 45 20 45 52 52 4F : 3A
DDA8 52 00 02 37 53 45 45 4B : B3
DDB0 20 45 52 52 4F 52 00 00 : AA
DDB8 37 4E 4F 54 20 53 41 4D : 29
DDC0 45 00 00 D0 00 00 00 00 : 15
DDC8 83 7D FB 0F FF 00 2C 10 : 45
DDD0 80 8D 00 C0 92 CF 87 83 : 38
DDD8 79 00 C0 FF 00 14 28 80 : F4
DDE0 8D FB 0F 92 CF 05 CF 87 : 53
DDE8 00 00 00 00 00 00 00 00 : 00
DDF0 00 00 00 00 00 00 00 00 : 00
DDF8 00 00 00 00 00 00 00 00 : 00
----------------------------------
SUM: 5F F6 D0 D3 11 FD 67 71 BDE1

DE00 00 00 00 00 00 00 00 00 : 00
DE08 00 00 00 00 00 00 00 00 : 00
DE10 00 00 00 00 00 00 00 00 : 00
DE18 00 00 00 00 00 00 00 00 : 00
DE20 00 00 00 00 00 00 00 00 : 00
DE28 00 00 00 00 00 00 00 00 : 00
DE30 00 00 00 00 00 00 00 00 : 00
DE38 00 00 00 00 00 00 00 00 : 00
DE40 00 00 00 00 00 00 00 00 : 00
DE48 00 00 00 00 00 00 00 00 : 00
DE50 00 00 00 00 00 00 00 00 : 00
DE58 00 00 00 00 00 00 00 00 : 00
DE60 00 00 00 00 00 00 00 00 : 00
DE68 00 00 20 20 20 20 4C 44 : 10
DE70 20 20 20 20 20 20 41 2C : 2D
DE78 43 20 20 20 20 20 20 20 : 23
----------------------------------
SUM: 63 40 60 60 60 60 AD 90 AB7F

DE80 20 20 20 20 20 3B 47 45 : 67
DE88 54 20 42 41 43 4B 20 53 : F8
DE90 54 41 54 55 53 0D 20 20 : DE
DE98 20 20 20 20 20 50 4F 50 : 8F
DEA0 20 20 20 20 20 42 43 20 : 45
DEA8 20 20 20 20 20 20 20 20 : 00
DEB0 20 20 20 20 3B 42 41 43 : 81
DEB8 4B 20 20 20 42 43 0D 20 : 5D
DEC0 20 20 20 20 20 20 52 45 : 57
DEC8 54 0D 0D 57 41 49 54 31 : D4
DED0 3A 20 20 20 20 20 20 20 : 1A
DED8 20 20 3B 57 41 49 54 20 : D0
DEE0 52 4F 55 54 49 4E 45 20 : 46
DEE8 32 0D 20 20 20 20 20 20 : FF
DEF0 20 4C 44 20 20 20 20 20 : 50
DEF8 20 41 2C 37 0D 57 41 49 : B2
----------------------------------
SUM: 25 77 C3 0F EB 81 67 0A 6B41
```

リスト3

```
0000                1 ;-----------------------------------------------------
0000                2          ;星が流れる      VER 3.3A
0000                3          ;                286 では動きません(ｽｽﾎﾞｹ)
0000                4 ;-----------------------------------------------------
D000                5          ORG      0D000H
D000                6
D000                7
D000                8 G_RAM:   EQU      0C000H
D000                9 HNM:              EQU      40       ;星の数
D000               10 MV:      EQU      3        ;(12)    ;速くしたいなら1に近づけて。
D000               11 FSP:     EQU      4        ;(16)    ;速くしたいなら1に近づけて。
D000               12
D000               13 P_INC:   EQU      15
D000               14 G_L:     EQU      0        ;G_RAM L
D000               15 G_H:     EQU      1        ;G_RAM H
D000               16 CNT:     EQU      2        ;COUNTER
D000               17 CNT_U:   EQU      3        ;USING
D000               18 SX1:     EQU      4
D000               19 SY1:     EQU      5
D000               20 SX2:     EQU      6
D000               21 SY2:     EQU      7
D000               22 SPD:     EQU      8        ;SPEED
D000               23 SPDM:    EQU      9
D000               24 SPD_P:   EQU      10       ;SPEED PLUS
D000               25 MVMNT:   EQU      11       ;移動距離
D000               26 DOT:     EQU      12       ;ﾄﾞｯﾄ形状
D000               27 XXX:     EQU      13
D000               28 YYY:     EQU      14
D000 CD 24 D0      29          CALL     ESTABLISH
D003 21 53 D1      30          LD       HL,INT_STAR
D006 22 5A 00      31          LD       (005AH),HL
D009 F3            32          DI
D00A ED 4B 20      33          LD       BC,(CTC0)
D00D D0
D00E 3E 58         34          LD       A,58H
D010 ED 79         35          OUT      (C),A
D012 ED 4B 22      36          LD       BC,(CTC1)
D015 D0
D016 3E A7         37          LD       A,167
D018 ED 79         38          OUT      (C),A
D01A 3E 80         39          LD       A,128
D01C ED 79         40          OUT      (C),A
D01E FB            41          EI
D01F C9            42          RET
D020               43
D020 A0 1F         44 CTC0:    DW       1FA0H             ;(704H)
D022 A1 1F         45 CTC1:    DW       1FA1H             ;(705H)
D024               46
D024               47 ESTABLISH:
D024 DD 21 C4      48          LD       IX,HBUF           ;立川君のOh! X LIVEみたいにこ
の場所に
D027 D2
D028 06 28         49          DO       B,HNM    {        ;いろんなメッセージでも書こう
かなと
D02A C5            50                   PUSH     BC       ;思ったけど音訓変換は非常に疲
れるからやめた。
D02B CD 37 D0      51                   CALL     DO_EST   ;久々のマクロ攻撃ソースリスト
をお楽しみください。
D02E 11 0F 00      52                   LD       DE,P_INC
D031 DD 19         53                   ADD      IX,DE
D033 C1            54                   POP      BC
D034 10 F4         55                   }
D036 C9            56          RET
D037               57
D037               58 DO_EST:
D037 CD 49 D2      59          CALL     RND
D03A 7D            60          LD       A,L
D03B B7            61          OR       A
D03C EA DC D0      62          JP       PE,OTHER          ;1/2の確率で分枝
D03F               63
D03F 11 33 00      64          LD       DE,51
D042 CD 87 D2      65          CALL     WARI              ;ANSWER DE = X
D045               66
D045 7B D6 80      67          IF       DE<640   THEN
D048 7A DE 02
D04B 30 04
D04D 3E F8         68                   LD       A,-8     ;SY1,2
D04F 18 0A         69          ELSE
D051 EB            70                   EX       DE,HL
D052 11 80 02      71                   LD       DE,640
D055 B7 ED 52      72                   SUB      HL,DE
D058 EB            73                   EX       DE,HL
D059 3E 08         74                   LD       A,8      ;SY1,2
D05B               75          F1
D05B 32 A3 D2      76          LD       (AWK),A
D05E 7B D6 40      77          IF       DE>=320 JR        RIGHT_SIDE
D061 7A DE 01
D064 30 3B
D066               78          ;        SX=1 DY=100
D066 21 40 01      79          LD       HL,320
D069 B7 ED 52      80          SUB      HL,DE    ;HL=320-X=DX
D06C E5            81          PUSH     HL
D06D 11 00 32      82          LD       DE,100*128
D070 EB            83          EX       DE,HL
D071 CD 87 D2      84          CALL     WARI
D074 DD 73 02      85          LD       (IX+CNT),E        ;INC/DEC COUNTER
D077 DD 36 06      86          LD       (IX+SX2),-1       ;SX2
D07A FF
D07B 3A A3 D2      87          LD       A,(AWK)
D07E DD 77 07      88          LD       (IX+SY2),A        ;SY2
D081 E1            89          POP      HL
D082 7D D6 64      90          IF       HL<100   THEN     ;DY>=DX
D085 7C DE 00
D088 30 0C
D08A DD 36 04      91                   LD       (IX+SX1),0        ;SX1
D08D 00
D08E 3A A3 D2      92                   LD       A,(AWK)
D091 DD 77 05      93                   LD       (IX+SY1),A        ;SY1
D094 18 08         94          ELSE                       ;DX>=DY
D096 DD 36 04      95                   LD       (IX+SX1),-1       ;SX1
D099 FF
D09A DD 36 05      96                   LD       (IX+SY1),0        ;SY1
D09D 00
D09E               97          FI
D09E C3 21 D1      98          JP       ETC_WK_SET
D0A1               99
D0A1              100 RIGHT_SIDE:       ;SX=-1 DY=100
D0A1 21 80 02     101          LD       HL,640
D0A4 B7 ED 52     102          SUB      HL,DE             ;HL=640-X=DX
D0A7 E5           103          PUSH     HL
D0A8 11 00 32     104          LD       DE,100*128
D0AB EB           105          EX       DE,HL
D0AC CD 87 D2     106          CALL     WARI
D0AF DD 73 02     107          LD       (IX+CNT),E
D0B2 DD 36 06     108          LD       (IX+SX2),1
D0B5 01
D0B6 3A A3 D2     109          LD       A,(AWK)
D0B9 DD 77 07     110          LD       (IX+SY2),A
D0BC E1           111          POP      HL
D0BD 7D D6 64     112          IF       HL<100 THEN       ;DY>=DX
D0C0 7C DE 00
D0C3 30 0C
D0C5 DD 36 04     113                   LD       (IX+SX1),0
D0C8 00
D0C9 3A A3 D2     114                   LD       A,(AWK)
D0CC DD 77 05     115                   LD       (IX+SY1),A
D0CF 18 08        116          ELSE
D0D1 DD 36 04     117                   LD       (IX+SX1),1
D0D4 01
D0D5 DD 36 05     118                   LD       (IX+SY1),0
D0D8 00
D0D9              119          FI
D0D9 C3 21 D1     120          JP       ETC_WK_SET
D0DC              121 OTHER:
D0DC 11 A4 00     122          LD       DE,164
D0DF CD 87 D2     123          CALL     WARI              ;ANSWER DE = Y
D0E2              124
D0E2 7B D6 C8     125          IF       DE<200   THEN
D0E5 7A DE 00
D0E8 30 04
D0EA 3E FF        126                   LD       A,-1     ;SX1,2
D0EC 18 0A        127          ELSE
D0EE EB           128                   EX       DE,HL
D0EF 11 C8 00     129                   LD       DE,200
D0F2 B7 ED 52     130                   SUB      HL,DE
D0F5 EB           131                   EX       DE,HL
D0F6 3E 01        132                   LD       A,1      ;SX1,2
D0F8              133          FI
D0F8 32 A3 D2     134          LD       (AWK),A
D0FB 7B           135          LD       A,E
D0FC FE 64 38     136          IF       A>=100   THEN
D0FF 04
D100 3E 08        137          LD       A,8
D102 18 02        138          ELSE
D104 3E F8        139          LD       A,-8
D106              140          FI
D106 F5           141          PUSH     AF
D107              142          ;SY=-1   DX=320
D107 21 00 A0     143          LD       HL,320*128
D10A CD 87 D2     144          CALL     WARI
D10D DD 73 02     145          LD       (IX+CNT),E        ;INC/DEC COUNTER
D110 3A A3 D2     146          LD       A,(AWK)
D113 DD 77 06     147          LD       (IX+SX2),A        ;SX2
D116 DD 77 04     148          LD       (IX+SX1),A        ;SX1
D119 F1           149          POP      AF
D11A DD 77 07     150          LD       (IX+SY2),A        ;SY2
D11D DD 36 05     151          LD       (IX+SY1),0        ;SY1
D120 00
D121              152
D121              153 MANNAKA:          EQU      23E8H+G_RAM      ;G_RAMのまんなかなので
す。ここが。
D121              154 MNK_H:   EQU      MANNAKA/256
D121              155 MNK_L:   EQU      -MNK_H*256+MANNAKA
D121              156
D121              157 ETC_WK_SET:
D121 DD 36 00     158          LD       (IX+G_L),MNK_L
D124 E8
D125 DD 36 01     159          LD       (IX+G_H),MNK_H
D128 E3
D129 DD 36 03     160          LD       (IX+CNT_U),0
D12C 00
D12D DD 36 0A     161          LD       (IX+SPD_P),FSP
D130 04
D131 DD 36 0B     162          LD       (IX+MVMNT),MV
D134 03
D135              163 AGAIN:
D135 CD 49 D2     164          CALL     RND
D138 7C           165          LD       A,H
D139 E6 07        166          AND      7
D13B 3C           167          INC      A
D13C DD 77 09     168          LD       (IX+SPDM),A
D13F DD 86 0A     169          ADD      A,(IX+SPD_P)
D142 DD 77 08     170          LD       (IX+SPD),A
D145 3E 80        171          LD       A,80H             ;10000000B
D147 DD 77 0C     172          LD       (IX+DOT),A
D14A DD 36 0D     173          LD       (IX+XXX),40
```


▶MusicBASICのボスコニアンFLASH FLASH FLASHは最高です。何年か前のイース組曲を聞くと西川さんの進歩が見える。みにくいアヒルの子が白鳥になったようなもんだ。ボスコニアン以外にも曲作ってください。　　赤松　宏章 (18) 兵庫県

```
D14D 28
D14E DD 36 0E  174          LD      (IX+YYY),12
D151 0C
D152 C9        175          RET
D153           176
D153           177 INT_STAR:
D153 F3        178          DI
D154 ED 73 0E  179          LD      (INTSP+1),SP
D157 D2
D158 31 C4 D2  180          LD      SP,USP
D15B F5        181          PUSH    AF
D15C C5        182          PUSH    BC
D15D D5        183          PUSH    DE
D15E E5        184          PUSH    HL
D15F DD E5     185          PUSH    IX
D161           186 MOVE:
D161 3E 28     187          LD      A,HNM           ;NUMBER OF HOSHI
D163 DD 21 C4  188          LD      IX,HBUF         ;HOSHI'S ADDRESS BUFFER
D166 D2
D167           189 MOVE_LP1:
D167 F5        190          PUSH    AF
D168           191
D168 DD 7E 08  192          LD      A,(IX+SPD)      ;SPEED COUNTER が 0にならなけ
れば
D16B 3D        193          DEC     A
D16C DD 77 08  194          LD      (IX+SPD),A
D16F C2 FD D1  195          JP      NZ,NEXT_HS      ;動かない。
D172 DD 4E 00  196          LD      C,(IX+G_L)
D175 DD 46 01  197          LD      B,(IX+G_H)      ;GET G RAM ADDRESS
D178 AF        198          XOR     A
D179 ED 79     199          OUT     (C),A           ;CLEAR DOT
D17B DD 7E 0B  200          LD      A,(IX+MVMNT)
D17E 3D        201          DEC     A
D17F DD 77 0B  202          LD      (IX+MVMNT),A            ;どのくらい移動したか
で
D182 20 0E     203          IF      Z       THEN
D184 3E 03     204                  LD      A,MV
D186 DD 77 0B  205                  LD      (IX+MVMNT),A
D189 DD 7E 0A  206                  LD      A,(IX+SPD_P)
D18C B7        207                  OR      A
D18D 28 03     208                  JR      Z,SPDSET
D18F DD 35 0A  209                  DEC     (IX+SPD_P)      ;SPEEDは変化
D192           210          FI
D192           211 SPDSET:
D192 DD 7E 09  212          LD      A,(IX+SPDM)     ;SPEED MASTER
D195 DD 86 0A  213          ADD     A,(IX+SPD_P)    ;SPEED PLUS
D198 DD 77 08  214          LD      (IX+SPD),A      ;SET NEW SPEED
D19B           215
D19B DD 7E 03  216          LD      A,(IX+CNT_U)
D19E DD 86 02  217          ADD     A,(IX+CNT)
D1A1 DD 77 03  218          LD      (IX+CNT_U),A
D1A4 FA AF D1  219          JP      M,SPECIAL
D1A7           220 ;NORMAL
D1A7 DD 5E 05  221          LD      E,(IX+SY1)
D1AA DD 56 04  222          LD      D,(IX+SX1)
D1AD 18 0A     223          JR      CALC
D1AF           224 SPECIAL:
D1AF DD 36 03  225          LD      (IX+CNT_U),0
D1B2 00
D1B3 DD 5E 07  226          LD      E,(IX+SY2)
D1B6 DD 56 06  227          LD      D,(IX+SX2)
D1B9           228 CALC:
D1B9 78        229          LD      A,B
D1BA 83        230          ADD     A,E
D1BB B7        231          OR      A
D1BC F2 13 D2  232          JP      P,PLUS
D1BF FE C0 DA  233          IF      A<0C0H  JP      MINUS
D1C2 2C D2
D1C4 47        234          LD      B,A
D1C5           235 SHIFT_DOT:
D1C5 7A        236          LD      A,D
D1C6 B7        237          OR      A
D1C7 FA DF D1  238          IF      P       THEN
D1CA DD CB 0C  239                  RRC     (IX+DOT)
D1CD 0E
D1CE 30 22     240                  JR      NC,WRT_DOT
D1D0 03        241                  INC     BC
D1D1 DD 7E 0D  242                  LD      A,(IX+XXX)
D1D4 3C        243                  INC     A
D1D5 FE 50 D2  244                  IF      A>=80   JP      NEW
D1D8 43 D2
D1DA DD 77 0D  245                  LD      (IX+XXX),A
D1DD 18 13     246          ELSE
D1DF DD CB 0C  247                  RLC     (IX+DOT)
D1E2 06
D1E3 30 0D     248                  JR      NC,WRT_DOT
D1E5 0B        249                  DEC     BC
D1E6 DD 7E 0D  250                  LD      A,(IX+XXX)
D1E9 3D        251                  DEC     A
D1EA FE FF CA  252                  IF      A=0FFH  JP      NEW
D1ED 43 D2
D1EF DD 77 0D  253                  LD      (IX+XXX),A
D1F2           254          FI
D1F2           255 WRT_DOT:
D1F2 DD 7E 0C  256          LD      A,(IX+DOT)
D1F5 ED 79     257          OUT     (C),A
D1F7           258
D1F7 DD 71 00  259          LD      (IX+G_L),C
D1FA DD 70 01  260          LD      (IX+G_H),B
D1FD           261 NEXT_HS:
D1FD 11 0F 00  262          LD      DE,P_INC
D200 DD 19     263          ADD     IX,DE
D202           264 NEXT_HS2:
D202 F1        265          POP     AF
D203 3D        266          DEC     A
D204 C2 67 D1  267          JP      NZ,MOVE_LP1
D207           268
D207 DD E1     269          POP     IX
D209 E1        270          POP     HL
D20A D1        271          POP     DE
D20B C1        272          POP     BC
D20C F1        273          POP     AF
D20D           274 INTSP:
D20D 31 00 00  275          LD      SP,0    ;DUMMY
D210 FB        276          EI
D211 ED 4D     277          RETI
D213           278
```

```
D213           279 PLUS:
D213 DD 7E 0E  280          LD      A,(IX+YYY)
D216 3C        281          INC     A
D217 FE 19 D2  282          IF      A>=25   JP      NEW
D21A 43 D2
D21C DD 77 0E  283          LD      (IX+YYY),A
D21F 60 69     284          LD      HL,BC
D221 01 B0 37  285          LD      BC,37B0H
D224 B7 ED 42  286          SUB     HL,BC
D227 44 4D     287          LD      BC,HL
D229 C3 C5 D1  288          JP      SHIFT_DOT
D22C           289
D22C           290 MINUS:
D22C DD 7E 0E  291          LD      A,(IX+YYY)
D22F 3D        292          DEC     A
D230 FE FF CA  293          IF      A=0FFH  JP      NEW
D233 43 D2
D235 DD 77 0E  294          LD      (IX+YYY),A
D238 60 69     295          LD      HL,BC
D23A 01 B0 37  296          LD      BC,37B0H
D23D 09        297          ADD     HL,BC
D23E 44 4D     298          LD      BC,HL
D240 C3 C5 D1  299          JP      SHIFT_DOT
D243           300
D243           301 NEW:
D243 CD 37 D0  302          CALL    DO_EST
D246 C3 02 D2  303          JP      NEXT_HS2
D249           304
D249           305 RND:
D249 ED 5B 5D  306          LD      DE,(OLDRND)
D24C D2
D24D ED 4B 60  307          LD      BC,(STEP)
D250 D2
D251 CD 62 D2  308          CALL    MULTI
D254 22 5D D2  309          LD      (OLDRND),HL
D257 ED 5F     310          LD      A,R
D259 32 5F D2  311          LD      (REFR),A
D25C C9        312          RET
D25D           313
D25D 23 E1     314 OLDRND: DW       0E123H
D25F 00        315 REFR:    DS      1
D260 E7 03     316 STEP:    DW      999
D262           317
D262           318 MULTI:
D262 21 00 00  319          LD      HL,0
D265 3E 10     320          LD      A,10H
D267           321 MLOOP:
D267 29        322          ADD     HL,HL
D268 CB 23     323          SLA     E
D26A CB 12     324          RL      D
D26C D2 70 D2  325          JP      NC,SKIP
D26F 09        326          ADD     HL,BC
D270           327 SKIP:
D270 3D        328          DEC     A
D271 C2 67 D2  329          JP      NZ,MLOOP
D274 C9        330          RET
D275           331
D275           332 KAKE: ; HL*DE  ｺﾀｴ=HL
D275 3E 10     333          LD      A,16
D277 44        334          LD      B,H
D278 4D        335          LD      C,L
D279 21 00 00  336          LD      HL,0
D27C           337 ML1:
D27C 29        338          ADD     HL,HL
D27D EB        339          EX      DE,HL
D27E 29        340          ADD     HL,HL
D27F EB        341          EX      DE,HL
D280 30 01     342          JR      NC,ML2
D282 09        343          ADD     HL,BC
D283           344 ML2:
D283 3D        345          DEC     A
D284 20 F6     346          JR      NZ,ML1
D286 C9        347          RET
D287           348
D287           349 WARI: ; HL/DE ｺﾀｴ= DE   ｱﾏﾘ= HL
D287 3E 10     350          LD      A,16
D289 42 4B     351          LD      BC,DE
D28B EB        352          EX      DE,HL
D28C 21 00 00  353          LD      HL,0
D28F           354
D28F           355 DV1:
D28F 29        356          ADD     HL,HL
D290 EB        357          EX      DE,HL
D291 29        358          ADD     HL,HL
D292 EB        359          EX      DE,HL
D293 30 01     360          JR      NC,DV2
D295 23        361          INC     HL
D296           362 DV2:
D296 B7 ED 42  363          SUB     HL,BC
D299 30 03     364          JR      NC,DV3
D29B 09        365          ADD     HL,BC
D29C 18 01     366          JR      DV4
D29E           367 DV3:
D29E 13        368          INC     DE
D29F           369 DV4:
D29F 3D        370          DEC     A
D2A0 20 ED     371          JR      NZ,DV1
D2A2 C9        372          RET
D2A3           373
D2A3 00        374 AWK:     DS      1
D2A4 00 00 00  375          DS      32
D2A7 00 00 00
D2AA 00 00 00
D2AD 00 00 00
D2B0 00 00 00
D2B3 00 00 00
D2B6 00 00 00
D2B9 00 00 00
D2BC 00 00 00
D2BF 00 00 00
D2C2 00 00
D2C4           376 USP:
D2C4           377 HBUF:
D2C3 [D2C3]
```

リスト4

```
0000            1 ;-----------------------
0000            2   ;DISK FORMAT & COPY
0000            3 ;-----------------------
0000            4        OFFSET  8000H-0D800H
D800            5        ORG     0D800H
D800            6
D800            7 D_AREA:        EQU     0DF00H
D800            8 VAR:           EQU     0FE00H  ;VARIABLE WORK AREA
D800            9 AD:            EQU     0DF00H  ;FORMAT DATA STARTS
D800 C3 0A D8  10        JP      COPY            ;DISK BACK UP
D803 C3 C4 D9  11        JP      FORMAT          ;DISK FORMAT
D806 C3 94 D8  12        JP      CHK_RD          ;CHECK READ
D809 00        13 STATUS:        DB      0
D80A           14 COPY:
D80A F3        15        DI
D80B ED 73 E8  16        LD      (SVSP),SP
D80E DD
D80F 31 6A DE  17        LD      SP,USP
D812 FB        18        EI
D813 AF        19        XOR     A
D814 32 B2 D9  20        LD      (SEKT),A
D817 11 B5 D9  21        LD      DE,RESTORE0
D81A CD E1 DB  22        CALL    DIOCS
D81D 11 BC D9  23        LD      DE,RESTORE1
D820 CD E1 DB  24        CALL    DIOCS
D823           25
D823 01 00 28  26        LD      BC,40*256       ;B=40 C=0
D826           27 COPY_LP0:
D826 C5        28        PUSH    BC
D827           29
D827 C5        30        PUSH    BC
D828 11 42 D9  31        LD      DE,MON0_SD0
D82B CD E1 DB  32        CALL    DIOCS           ;MOTOR ON DRIVE 0
D82E C1        33        POP     BC
D82F           34
D82F AF        35        XOR     A
```

```
D830 32 C3 D9  36        LD      (DRIVE),A
D833 3A B2 D9  37        LD      A,(SEKT)
D836 32 B3 D9  38        LD      (LAST),A
D839 79        39        LD      A,C
D83A 32 B2 D9  40        LD      (SEKT),A
D83D CD C0 DB  41        CALL    WHERE
D840 11 B0 D9  42        LD      DE,SKCOM
D843 CD E1 DB  43        CALL    DIOCS           ;SEEK DRIVE0
D846           44
D846 21 00 DF  45        LD      HL,D_AREA
D849 22 C2 DD  46        LD      (BUFAD),HL
D84C 11 4E D9  47        LD      DE,R_COM
D84F CD E1 DB  48        CALL    DIOCS           ;READ SIDE0
D852           49
D852 11 45 D9  50        LD      DE,MON0_SD1
D855 CD E1 DB  51        CALL    DIOCS           ;SIDE1
D858           52
D858 11 4E D9  53        LD      DE,R_COM
D85B CD E1 DB  54        CALL    DIOCS           ;READ SIDE1
D85E           55
D85E 11 48 D9  56        LD      DE,MON1_SD0
D861 CD E1 DB  57        CALL    DIOCS           ;MOTOR ON DRIVE 1
D864           58
D864 3E 01     59        LD      A,1
D866 32 C3 D9  60        LD      (DRIVE),A
D869           61
D869 11 B0 D9  62        LD      DE,SKCOM
D86C CD E1 DB  63        CALL    DIOCS           ;SEEK DRIVE1
D86F           64
D86F 21 00 DF  65        LD      HL,D_AREA
D872 22 C2 DD  66        LD      (BUFAD),HL
D875 11 7F D9  67        LD      DE,W_COM
D878 CD E1 DB  68        CALL    DIOCS           ;WRITE SIDE0
D87B           69
D87B 11 4B D9  70        LD      DE,MON1_SD1
D87E CD E1 DB  71        CALL    DIOCS           ;SIDE1
```

▶BASICとPASCALをちょっとかじったんですが，やっぱ，どーもアセンブラのほうが性にあってるような気がします。　　丸山　勝之（20）埼玉県

```
D881                72
D881 11 7F D9       73          LD      DE,W_COM
D884 CD E1 DB       74          CALL    DIOCS             ;WRITE SIDE1
D887                75
D887 C1             76          POP     BC
D888 0C             77          INC     C
D889 10 9B          78          DJNZ    COPY_LP0
D88B 11 3D D9       79          LD      DE,M_OFF
D88E CD E1 DB       80          CALL    DIOCS
D891 C3 67 DD       81          JP      EXIT
D894                82
D894                83 CHK_RD:
D894 F3             84          DI
D895 ED 73 E8       85          LD      (SVSP),SP
D898 DD
D899 31 6A DE       86          LD      SP,USP
D89C FB             87          EI
D89D AF             88          XOR     A
D89E 32 B2 D9       89          LD      (SEKT),A
D8A1 11 B5 D9       90          LD      DE,RESTORE0
D8A4 CD E1 DB       91          CALL    DIOCS
D8A7 11 BC D9       92          LD      DE,RESTORE1
D8AA CD E1 DB       93          CALL    DIOCS
D8AD                94
D8AD 01 00 28       95          LD      BC,40*256         ;B=40 C=0
D8B0                96 CHCK_LP0:
D8B0 C5             97          PUSH    BC
D8B1                98
D8B1 C5             99          PUSH    BC
D8B2 11 42 D9      100          LD      DE,MON0_SD0
D8B5 CD E1 DB      101          CALL    DIOCS             ;MOTOR ON DRIVE 0
D8B8 C1            102          POP     BC
D8B9               103
D8B9 AF            104          XOR     A
D8BA 32 C3 D9      105          LD      (DRIVE),A
D8BD 3A B2 D9      106          LD      A,(SEKT)
D8C0 32 B3 D9      107          LD      (LAST),A
D8C3 79            108          LD      A,C
D8C4 32 B2 D9      109          LD      (SEKT),A
D8C7 CD C0 DB      110          CALL    WHERE
D8CA 11 B0 D9      111          LD      DE,SKCOM
D8CD CD E1 DB      112          CALL    DIOCS             ;SEEK DRIVE0
D8D0               113
D8D0 21 00 DF      114          LD      HL,D_AREA
D8D3 22 C2 DD      115          LD      (BUFAD),HL
D8D6 11 4E D9      116          LD      DE,R_COM
D8D9 CD E1 DB      117          CALL    DIOCS             ;READ SIDE0
D8DC               118
D8DC 11 45 D9      119          LD      DE,MON0_SD1
D8DF CD E1 DB      120          CALL    DIOCS             ;SIDE1
D8E2               121
D8E2 11 4E D9      122          LD      DE,R_COM
D8E5 CD E1 DB      123          CALL    DIOCS             ;READ SIDE1
D8E8               124
D8E8 CD 2A D9      125          CALL    CALC_SUM
D8EB F5            126          PUSH    AF
D8EC               127
D8EC 11 48 D9      128          LD      DE,MON1_SD0
D8EF CD E1 DB      129          CALL    DIOCS             ;MOTOR ON DRIVE 1
D8F2               130
D8F2 3E 01         131          LD      A,1
D8F4 32 C3 D9      132          LD      (DRIVE),A
D8F7               133
D8F7 11 B0 D9      134          LD      DE,SKCOM
D8FA CD E1 DB      135          CALL    DIOCS             ;SEEK DRIVE1
D8FD               136
D8FD 21 00 DF      137          LD      HL,D_AREA
D900 22 C2 DD      138          LD      (BUFAD),HL
D903 11 4E D9      139          LD      DE,R_COM
D906 CD E1 DB      140          CALL    DIOCS             ;READ SIDE0
D909               141
D909 11 4B D9      142          LD      DE,MON1_SD1
D90C CD E1 DB      143          CALL    DIOCS             ;SIDE1
D90F               144
D90F 11 4E D9      145          LD      DE,R_COM
D912 CD E1 DB      146          CALL    DIOCS             ;READ SIDE1
D915               147
D915 CD 2A D9      148          CALL    CALC_SUM
D918 E1            149          POP     HL
D919 BC            150          CP      H
D91A C2 30 DD      151          JP      NZ,ERROR4         ;データが違う
D91D               152
D91D C1            153          POP     BC
D91E 0C            154          INC     C
D91F 10 8F         155          DJNZ    CHCK_LP0
D921 11 3D D9      156          LD      DE,M_OFF
D924 CD E1 DB      157          CALL    DIOCS
D927 C3 67 DD      158          JP      EXIT
D92A               159
D92A               160 CALC_SUM:
D92A 21 00 DF      161          LD      HL,D_AREA
D92D 11 00 20      162          LD      DE,BYTES*SCTR*2
D930 0E 00         163          LD      C,0
D932               164 SUM_LP0:
D932 79            165          LD      A,C
D933 AE            166          XOR     (HL)
D934 23            167          INC     HL
D935 1B            168          DEC     DE
D936 4F            169          LD      C,A
D937 7A            170          LD      A,D
D938 B3            171          OR      E
D939 20 F7         172          JR      NZ,SUM_LP0
D93B 79            173          LD      A,C
D93C C9            174          RET
D93D               175 M_OFF:
D93D 04 00 04      176          DB      4,0,4,1,0FFH
D940 01 FF
D942               177 MON0_SD0:
D942 04 80 FF      178          DB      4,80H,0FFH
D945               179 MON0_SD1:
D945 04 90 FF      180          DB      4,90H,0FFH
D948               181 MON1_SD0:
D948 04 81 FF      182          DB      4,81H,0FFH
D94B               183 MON1_SD1:
D94B 04 91 FF      184          DB      4,91H,0FFH
D94E               185 R_COM:           ;READ FROM DRIVE 0
D94E               186          ;READ COM,START SECTOR
D94E 02 80 01      187          DB      2,80H,1,2,80H,2,2,80H,3,2,80H,4
D951 02 80 02
D954 02 80 03
D957 02 80 04
D95A 02 80 05      188          DB      2,80H,5,2,80H,6,2,80H,7,2,80H,8
D95D 02 80 06
D960 02 80 07
D963 02 80 08
D966 02 80 09      189          DB      2,80H,9,2,80H,10,2,80H,11,2,80H,12
D969 02 80 0A
D96C 02 80 0B
D96F 02 80 0C
D972 02 80 0D      190          DB      2,80H,13,2,80H,14,2,80H,15,2,80H,16
D975 02 80 0E
D978 02 80 0F
D97B 02 80 10
D97E FF            191          DB      0FFH    ;END CODE
D97F               192
D97F               193 W_COM:           ;WRITE TO DRIVE 1
D97F               194          ;WRITE COM,START SECTOR
D97F 03 A0 01      195          DB      3,0A0H,1,3,0A0H,2,3,0A0H,3,3,0A0H,4
D982 03 A0 02
D985 03 A0 03
D988 03 A0 04
D98B 03 A0 05      196          DB      3,0A0H,5,3,0A0H,6,3,0A0H,7,3,0A0H,8
D98E 03 A0 06
D991 03 A0 07
D994 03 A0 08
D997 03 A0 09      197          DB      3,0A0H,9,3,0A0H,10,3,0A0H,11,3,0A0H,12
D99A 03 A0 0A
D99D 03 A0 0B
D9A0 03 A0 0C
D9A3 03 A0 0D      198          DB      3,0A0H,13,3,0A0H,14,3,0A0H,15,3,0A0H,16
D9A6 03 A0 0E
D9A9 03 A0 0F
D9AC 03 A0 10
D9AF FF            199          DB      0FFH    ;END CODE
D9B0               200
D9B0               201 SKCOM:           ;SEEK TRACK
D9B0 01 1C         202          DB      1,1CH
D9B2 00            203 SEKT:    DB      0
D9B3 00            204 LAST:    DB      0
D9B4 FF            205          DB      0FFH    ;END CODE
D9B5               206 RESTORE0:
D9B5 04 80 00      207          DB      4,80H,0,0,4,0,0FFH
D9B8 00 04 00
D9BB FF
D9BC               208 RESTORE1:
D9BC 04 81 00      209          DB      4,81H,0,0,4,1,0FFH
D9BF 00 04 01
D9C2 FF
D9C3               210
D9C3 00            211 DRIVE:           DB      0
D9C4               212
D9C4               213 SCTR:   EQU     16
D9C4               214 END:    EQU     39
D9C4               215 DSM:    EQU     0FFCH             ;DSMといってもKORGのデジタルサンプ
ラーとは関係ない。
D9C4               216 DTYPE:  EQU     0FFFH
D9C4               217 FORMAT:
D9C4 F3            218          DI
D9C5 ED 73 E8      219          LD      (SVSP),SP
D9C8 DD
D9C9 31 6A DE      220          LD      SP,USP
D9CC FB            221          EI
D9CD 21 00 DF      222          LD      HL,AD
D9D0 22 C2 DD      223          LD      (BUFAD),HL
D9D3 22 AC DB      224          LD      (KAD),HL
D9D6 22 B1 DB      225          LD      (CPYAD),HL
D9D9               226 ;FORMAT MAIN
D9D9 CD B1 DA      227          CALL    MAKE_F
D9DC               228
D9DC 01 FF 0F      229          LD      BC,DTYPE          ;2D/2DD  MODE
D9DF ED 78         230          IN      A,(C)
D9E1               231
D9E1 11 BC D9      232          LD      DE,RESTORE1
D9E4 CD E1 DB      233          CALL    DIOCS             ;DRIVE 1 MOTOR ON
D9E7               234
D9E7 AF            235          XOR     A
D9E8 32 AA DB      236          LD      (CYL),A
D9EB 32 AB DB      237          LD      (OLD),A
D9EE               238
D9EE               239 FMTRP0:
D9EE CD 4C DB      240          CALL    SEEK#
D9F1 3A AA DB      241          LD      A,(CYL)
D9F4 CD C0 DB      242          CALL    WHERE
D9F7 32 AB DB      243          LD      (OLD),A
D9FA               244
D9FA AF            245          XOR     A
D9FB 32 A9 DB      246          LD      (SIDE),A
D9FE               247 SDRP0:
D9FE               248
D9FE 01 FC 0F      249          LD      BC,DSM
DA01 3A A9 DB      250          LD      A,(SIDE)
DA04 FE 01         251          CP      1
DA06 20 04         252          JR      NZ,SIDE1
DA08 1E 10         253          LD      E,10H
DA0A 18 02         254          JR      DO_FMT
DA0C               255 SIDE1:
DA0C 1E 00         256          LD      E,0
DA0E               257 DO_FMT:
DA0E 3E 81         258          LD      A,81H
DA10 B3            259          OR      E
DA11 ED 79         260          OUT     (C),A
DA13 CD CC DC      261          CALL    WNBSY
DA16               262
DA16 CD 14 DB      263          CALL    ST_FMT
DA19               264
DA19               265
DA19 16 11         266          LD      D,17
DA1B 21 AF DB      267          LD      HL,DMAWRT
DA1E CD F7 DC      268          CALL    SETDMA
DA21 01 F8 0F      269          LD      BC,CR
DA24 3E F0         270          LD      A,0F0H
DA26 ED 79         271          OUT     (C),A
DA28 CD CC DC      272          CALL    WNBSY
DA2B 16 01         273          LD      D,1
DA2D 21 AF DB      274          LD      HL,DMAWRT
DA30 CD F7 DC      275          CALL    SETDMA
DA33               276
DA33 3A A9 DB      277          LD      A,(SIDE)
DA36 3C            278          INC     A
DA37 32 A9 DB      279          LD      (SIDE),A
DA3A FE 01         280          CP      1
DA3C 28 C0         281          JR      Z,SDRP0
DA3E 3A AA DB      282          LD      A,(CYL)
DA41 3C            283          INC     A
DA42 32 AA DB      284          LD      (CYL),A
DA45 FE 28         285          CP      END+1
DA47 20 A5         286          JR      NZ,FMTRP0
DA49               287 ;MOTOR OFF
DA49 01 FC 0F      288          LD      BC,DSM
DA4C 3E 01         289          LD      A,1
DA4E ED 79         290          OUT     (C),A
DA50               291
DA50               292          ;SYSTEM INITIALIZE
DA50               293
DA50 11 01 DF      294          LD      DE,D_AREA+1       ;システムフォーマットデータを
作る。
DA53 62            295          LD      H,D
DA54 6B            296          LD      L,E
DA55 2B            297          DEC     HL
DA56 36 FF         298          LD      (HL),0FFH
DA58 01 FF 1F      299          LD      BC,2000H-1
DA5B ED B0         300          LDIR
DA5D 11 01 ED      301          LD      DE,256*14+D_AREA+1
DA60 62            302          LD      H,D
DA61 6B            303          LD      L,E
DA62 2B            304          DEC     HL
DA63 36 00         305          LD      (HL),0
DA65 01 FF 01      306          LD      BC,256*2-1
DA68 ED B0         307          LDIR
DA6A 21 00 ED      308          LD      HL,256*14+D_AREA
DA6D 36 01         309          LD      (HL),1
DA6F 23            310          INC     HL
DA70 36 8F         311          LD      (HL),8FH
DA72               312
DA72 11 51 ED      313          LD      DE,256*14+D_AREA+$50+1
DA75 62            314          LD      H,D
DA76 6B            315          LD      L,E
DA77 2B            316          DEC     HL
DA78 36 8F         317          LD      (HL),8FH
DA7A 01 2F 00      318          LD      BC,16*3-1
DA7D ED B0         319          LDIR
DA7F               320
DA7F 11 BC D9      321          LD      DE,RESTORE1
DA82 CD E1 DB      322          CALL    DIOCS
DA85               323
DA85 11 48 D9      324          LD      DE,MON1_SD0
DA88 CD E1 DB      325          CALL    DIOCS             ;SIDE0 (DRIVE1)
DA8B               326
DA8B 3E 01         327          LD      A,1
DA8D 32 C3 D9      328          LD      (DRIVE),A
DA90               329
DA90 21 00 DF      330          LD      HL,D_AREA
DA93 22 C2 DD      331          LD      (BUFAD),HL
DA96 11 7F D9      332          LD      DE,W_COM
DA99 CD E1 DB      333          CALL    DIOCS             ;SYSTEM FORMAT DATA を書く
DA9C               334
DA9C 11 4B D9      335          LD      DE,MON1_SD1
DA9F CD E1 DB      336          CALL    DIOCS             ;SIDE1 (DRIVE1)
DAA2               337
DAA2 11 7F D9      338          LD      DE,W_COM
DAA5 CD E1 DB      339          CALL    DIOCS             ;SYSTEM FORMAT DATA を書く そ
の2
DAA8               340
DAA8 11 3D D9      341          LD      DE,M_OFF
DAAB CD E1 DB      342          CALL    DIOCS
DAAE               343
DAAE C3 67 DD      344          JP      EXIT
DAB1               345
DAB1               346 MAKE_F:
DAB1 DD 21 00      347          LD      IX,VAR
DAB4 FE
DAB5 DD 36 00      348          LD      (IX+0),0
DAB8 00
DAB9               349
DAB9 21 67 DB      350          LD      HL,GAP1
DABC CD D2 DA      351          CALL    MFSUB
DABF               352
DABF 06 10         353          LD      B,SCTR            ;NUMBER OF SECTORS
```

▶X1の記事が減ったな……俺がプログラムを送ればひとつ増えるかもしれないんだな。
和気　資明 (18) 広島県

```
DAC1            354 FMT_LP:
DAC1 C5         355         PUSH    BC
DAC2 21 6C DB   356         LD      HL,SECTOR
DAC5 CD D2 DA   357         CALL    MFSUB
DAC8 C1         358         POP     BC
DAC9 10 F6      359         DJNZ    FMT_LP
DACB            360
DACB 21 92 DB   361         LD      HL,GAP4
DACE CD D2 DA   362         CALL    MFSUB
DAD1 C9         363         RET
DAD2            364
DAD2            365 MFSUB:
DAD2 22 A7 DB   366         LD      (KHL),HL
DAD5            367 MFLOOP:
DAD5 2A A7 DB   368         LD      HL,(KHL)
DAD8 4E         369         LD      C,(HL)
DAD9 23         370         INC     HL
DADA 46         371         LD      B,(HL)
DADB 23         372         INC     HL
DADC 78         373         LD      A,B
DADD B1         374         OR      C
DADE C8         375         RET     Z
DADF 7E         376         LD      A,(HL)
DAE0 23         377         INC     HL
DAE1 22 A7 DB   378         LD      (KHL),HL
DAE4 FE FF      379         CP      0FFH
DAE6 28 03      380         JR      Z,SET_ID_ADR
DAE8 57         381         LD      D,A
DAE9 18 1A      382         JR      MF2
DAEB            383 SET_ID_ADR:
DAEB DD 34 00   384         INC     (IX+0)
DAEE 16 00      385         LD      D,0
DAF0 DD 5E 00   386         LD      E,(IX+0)
DAF3 CB 23      387         SLA     E
DAF5 DD E5      388         PUSH    IX
DAF7 DD 19      389         ADD     IX,DE
DAF9            390
DAF9 ED 5B AC   391         LD      DE,(KAD)
DAFC DB
DAFD DD 73 00   392         LD      (IX+0),E
DB00 DD 72 01   393         LD      (IX+1),D                ;ID ADDRESS
DB03            394                                         ;後でこのADDRESSに実際のIDを書
く。
DB03 DD E1      395         POP     IX
DB05            396
DB05            397 MF2:
DB05 2A AC DB   398         LD      HL,(KAD)
DB08            399 MF2_LP:
DB08 72         400         LD      (HL),D
DB09 23         401         INC     HL
DB0A 0B         402         DEC     BC
DB0B 78         403         LD      A,B
DB0C B1         404         OR      C
DB0D 20 F9      405         JR      NZ,MF2_LP
DB0F 22 AC DB   406         LD      (KAD),HL
DB12 18 C1      407         JR      MFLOOP
DB14            408
DB14            409 ST_FMT:                         ;IDをセット
DB14 3E 01      410         LD      A,1
DB16 32 AE DB   411         LD      (PPP),A
DB19 DD 4E 00   412         LD      C,(IX)
DB1C            413 LP01:
DB1C DD E5      414         PUSH    IX
DB1E 3A AE DB   415         LD      A,(PPP)
DB21 16 00      416         LD      D,0
DB23 5F         417         LD      E,A
DB24 CB 23      418         SLA     E
DB26 DD 19      419         ADD     IX,DE
DB28 DD 6E 00   420         LD      L,(IX+0)
DB2B DD 66 01   421         LD      H,(IX+1)
DB2E DD E1      422         POP     IX
DB30 3A AA DB   423         LD      A,(CYL)                 ;ｼﾘﾝﾀﾞｰ
DB33 77         424         LD      (HL),A
DB34 23         425         INC     HL
DB35 3A A9 DB   426         LD      A,(SIDE)                ;ｻｲﾄﾞ(裏表)
DB38 77         427         LD      (HL),A
DB39 23         428         INC     HL
DB3A 3A AE DB   429         LD      A,(PPP)                 ;ｾｸﾀｰ番号
DB3D 77         430         LD      (HL),A
DB3E 23         431         INC     HL
DB3F            432
DB3F 36 01      433         LD      (HL),1                  ;ﾚﾝｸﾞｽ長
DB41 3A AE DB   434         LD      A,(PPP)
DB44 B9         435         CP      C
DB45 D0         436         RET     NC
DB46 3C         437         INC     A
DB47 32 AE DB   438         LD      (PPP),A
DB4A 18 D0      439         JR      LP01
DB4C            440 SEEK#:
DB4C 01 FB 0F   441         LD      BC,DR
DB4F 3A AA DB   442         LD      A,(CYL)
DB52 ED 79      443         OUT     (C),A
DB54            444
DB54 01 F9 0F   445         LD      BC,TR
DB57 3A AB DB   446         LD      A,(OLD)
DB5A ED 79      447         OUT     (C),A
DB5C            448
DB5C 01 F8 0F   449         LD      BC,CR
DB5F 3E 1C      450         LD      A,1CH                   ;SEEK
DB61 ED 79      451         OUT     (C),A
DB63 CD CC DC   452         CALL    WNBSY
DB66 C9         453         RET
DB67            454
DB67            455 ;FORMAT DATA
DB67            456 GAP1:
DB67 20 00 4E   457         DW      32      DB      4EH
DB6A 00 00      458         DW      0
DB6C            459 SECTOR:
DB6C 0C 00 00   460         DW      12      DB      0
DB6F 03 00 F5   461         DW      3       DB      0F5H
DB72 01 00 FE   462         DW      1       DB      0FEH
DB75 04 00 FF   463         DW      4       DB      0FFH
DB78 01 00 F7   464         DW      1       DB      0F7H
DB7B 16 00 4E   465         DW      22      DB      4EH
DB7E 0C 00 00   466         DW      12      DB      0
DB81 03 00 F5   467         DW      3       DB      0F5H
DB84 01 00 FB   468         DW      1       DB      0FBH
DB87 00 01 E5   469         DW      256     DB      0E5H
DB8A 01 00 F7   470         DW      1       DB      0F7H
DB8D 36 00 4E   471         DW      54      DB      4EH
DB90 00 00      472         DW      0
DB92            473 GAP4
DB92 0A 01 4E   474         DW      266     DB      4EH
DB95 00 00      475         DW      0
DB97            476 SQUE
DB97 01 02 03   477         DB      1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
DB9A 04 05 06
DB9D 07 08 09
DBA0 0A 0B 0C
DBA3 0D 0E 0F
DBA6 10
DBA7 00 00      478 KHL:    DW      0
DBA9 00         479 SIDE:   DB      0
DBAA 00         480 CYL:    DB      0
DBAB 00         481 OLD:    DB      0
DBAC 00 00      482 KAD:    DW      0
DBAE 00         483 PPP:    DB      0
DBAF            484
DBAF            485 DMAWRT:         ;17 BYTES
DBAF 83 79      486         DB      83H,79H
DBB1            487 CPYAD:
DBB1 00 00      488         DB      0,0
DBB3            489
DBB3 AF 28 14   490         DB      0AFH,28H,14H,28H,80H,8DH,0FBH,0FH,92H,0CFH,05H
,0CFH,87H
DBB6 28 80 8D
DBB9 FB 0F 92
DBBC CF 05 CF
DBBF 87
DBC0            491
DBC0            492 TEXT:   EQU     80*20+3000H
DBC0            493
DBC0            494
DBC0            495 WHERE:
DBC0 F5         496         PUSH    AF
DBC1 C5         497         PUSH    BC
DBC2 E5         498         PUSH    HL
DBC3 87         499         ADD     A,A
DBC4 21 40 36   500         LD      HL,TEXT
DBC7 06 00      501         LD      B,0
DBC9 4F         502         LD      C,A
DBCA 09         503         ADD     HL,BC
DBCB 4D         504         LD      C,L
DBCC 44         505         LD      B,H
DBCD 3E 87      506         LD      A,87H                   ;四角
DBCF ED 79      507         OUT     (C),A
DBD1 03         508         INC     BC
DBD2 ED 79      509         OUT     (C),A
DBD4            510
DBD4 3E 87      511         LD      A,7+128
DBD6 CB A0      512         RES     4,B
DBD8 ED 79      513         OUT     (C),A
DBDA 0B         514         DEC     BC
DBDB ED 79      515         OUT     (C),A
DBDD            516
DBDD E1         517         POP     HL
DBDE C1         518         POP     BC
DBDF F1         519         POP     AF
DBE0 C9         520         RET
DBE1            521
DBE1            522 ;-------------------------------------------------
DBE1            523         ;DMA DISK IOCS
DBE1            524         ;SPECIAL THANKS TO 祝一平著
DBE1            525         ;                       「試験に出るX1」
DBE1            526 ;-------------------------------------------------
DBE1            527
DBE1            528 CR:     EQU     0FF8H   ;COM    REG
DBE1            529 STR:    EQU     0FF8H   ;STAT   REG
DBE1            530 TR:     EQU     0FF9H   ;TRACK  REG
DBE1            531 SCR:    EQU     0FFAH   ;SECTOR REG
DBE1            532 DR:     EQU     0FFBH   ;DATA   REG
DBE1            533
DBE1            534 DIOCS:
DBE1            535         ;ENTRY
DBE1            536         ;DE = COMMAND START ADDRESS
DBE1 ED 53 C6   537         LD      (KEEPCOM),DE
DBE4 DD
DBE5            538 START
DBE5 1A         539         LD      A,(DE)          ;GET    COMM No.
DBE6 FE FF      540         CP      0FFH
DBE8 C8         541         RET     Z
DBE9 13         542         INC     DE
DBEA ED 53 C4   543         LD      (KDE),DE
DBED DD
DBEE 21 08 DC   544         LD      HL,JTA          ;HL=JUMP TABLE AREA
DBF1 16 00      545         LD      D,0
DBF3 87         546         ADD     A,A
DBF4 5F         547         LD      E,A             ;DE=A*2
DBF5 19         548         ADD     HL,DE
DBF6 5E         549         LD      E,(HL)
DBF7 23         550         INC     HL
DBF8 56         551         LD      D,(HL)          ;GET JUMP ADD
DBF9 EB         552         EX      DE,HL           ; HL=JUMP ADD
DBFA ED 5B C4   553         LD      DE,(KDE)
DBFD DD
DBFE            554
DBFE CD 07 DC   555         CALL    PATCH           ;CALL   (HL)
DC01            556
DC01 ED 5B C4   557         LD      DE,(KDE)
DC04 DD
DC05 18 DE      558         JR      START
DC07            559 PATCH:
DC07 E9         560         JP      (HL)
DC08            561
DC08            562 ;TYPE I COM
DC08            563 JTA:
DC08 12 DC      564         DW      RSTR            ;RESTORE        0
DC0A 21 DC      565         DW      SEEK            ;SEEK           1
DC0C            566
DC0C            567 ;TYPE II COM
DC0C 4B DC      568         DW      READD           ;READ   DATA    2
DC0E 83 DC      569         DW      WRITD           ;WRITE  DATA    3
DC10            570
DC10            571 ;MOTOR,SIDE,DRIVE
DC10 BB DC      572         DW      MSD             ;               4
DC12            573
DC12            574 RSTR:
DC12 1A         575         LD      A,(DE)
DC13 13         576         INC     DE
DC14 ED 53 C4   577         LD      (KDE),DE
DC17 DD
DC18 01 F8 0F   578         LD      BC,CR           ;COMM REG
DC1B ED 79      579         OUT     (C),A
DC1D CD CC DC   580         CALL    WNBSY           ;WAIT
DC20 C9         581         RET
DC21            582
DC21            583 SEEK:
DC21 EB         584         EX      DE,HL
DC22 7E         585         LD      A,(HL)          ;A=COM
DC23 23         586         INC     HL
DC24 56         587         LD      D,(HL)          ;D=TRACK# TO SEEK
DC25 23         588         INC     HL
DC26 5E         589         LD      E,(HL)          ;E=CURRENT TRACK#
DC27 23         590         INC     HL
DC28 22 C4 DD   591         LD      (KDE),HL
DC2B 01 F9 0F   592         LD      BC,TR
DC2E ED 59      593         OUT     (C),E           ;SET CURRENT TRACK#
DC30 01 FB 0F   594         LD      BC,DR
DC33 ED 51      595         OUT     (C),D           ;SET TRACK TO SEEK
DC35 01 F8 0F   596         LD      BC,CR
DC38 ED 79      597         OUT     (C),A           ;SEND   SEEK COM
DC3A CD CC DC   598         CALL    WNBSY
DC3D 01 F8 0F   599         LD      BC,STR
DC40 ED 78      600         IN      A,(C)
DC42 32 09 D8   601         LD      (STATUS),A
DC45 E6 18      602         AND     18H             ;00011000B
DC47 C2 24 DD   603         JP      NZ,ERROR3          ;(SEEK ERROR)
DC4A C9         604         RET
DC4B            605
DC4B            606 READD:
DC4B D9         607         EXX
DC4C 2A C2 DD   608         LD      HL,(BUFAD)
DC4F 22 D2 DD   609         LD      (RD_AD),HL
DC52 01 00 01   610         LD      BC,BYTES
DC55 09         611         ADD     HL,BC
DC56 22 C2 DD   612         LD      (BUFAD),HL
DC59 21 C8 DD   613         LD      HL,R_DMA
DC5C 16 0F      614         LD      D,15
DC5E CD F7 DC   615         CALL    SETDMA
DC61 D9         616         EXX
DC62 CD E4 DC   617         CALL    SETSCT
DC65 01 F8 0F   618         LD      BC,CR
DC68 ED 79      619         OUT     (C),A
DC6A CD CC DC   620         CALL    WNBSY
DC6D            621 ;RESET
DC6D 21 C8 DD   622         LD      HL,R_DMA
DC70 16 01      623         LD      D,1
DC72 CD F7 DC   624         CALL    SETDMA          ;83HのみDMAへ出力 = RESET
DC75            625
DC75 01 F8 0F   626         LD      BC,STR
DC78 ED 78      627         IN      A,(C)
DC7A 32 09 D8   628         LD      (STATUS),A
DC7D E6 1C      629         AND     1CH             ;00011100B
DC7F C8         630         RET     Z
DC80 C3 01 DD   631         JP      ERROR
DC83            632
DC83            633 WRITD:
DC83 D9         634         EXX
DC84 2A C2 DD   635         LD      HL,(BUFAD)
DC87 22 D9 DD   636         LD      (WR_AD),HL
DC8A 01 00 01   637         LD      BC,BYTES
DC8D 09         638         ADD     HL,BC
DC8E 22 C2 DD   639         LD      (BUFAD),HL
DC91 21 D7 DD   640         LD      HL,W_DMA
DC94 16 11      641         LD      D,17
DC96 CD F7 DC   642         CALL    SETDMA
DC99 D9         643         EXX
DC9A CD E4 DC   644         CALL    SETSCT
DC9D 01 F8 0F   645         LD      BC,CR
DCA0 ED 79      646         OUT     (C),A
DCA2 CD CC DC   647         CALL    WNBSY
DCA5            648 ;RESET
DCA5 21 D7 DD   649         LD      HL,W_DMA
DCA8 16 01      650         LD      D,1
DCAA CD F7 DC   651         CALL    SETDMA          ;83HのみDMAへ出力 = RESET
DCAD            652
DCAD 01 F8 0F   653         LD      BC,STR
DCB0 ED 78      654         IN      A,(C)
```

▶やっと，ジェノサイドをクリアできました（試験期間中）。やっぱ，勉強に疲れたときはマーダークラブDXのエンディングを見て気分転換ですね。　小林　毅（22）新潟県

```
DCB2 32 09 D8  655          LD     (STATUS),A
DCB5 E6 3C     656          AND    3CH              ;00111100B
DCB7 C8        657          RET    Z
DCB8 C3 18 DD  658          JP     ERROR2
DCBB           659
DCBB           660 MSD:
DCBB 1A        661          LD     A,(DE)           ;GET DATA
DCBC 13        662          INC    DE
DCBD ED 53 C4  663          LD     (KDE),DE
DCC0 DD
DCC1 01 FC 0F  664          LD     BC,DSM
DCC4 ED 79     665          OUT    (C),A            ;SET IT
DCC6 B7        666          OR     A                ;MOTOR ON?
DCC7 F0        667          RET    P                ;MOTOR OFF THEN RET
DCC8 CD CC DC  668          CALL   WNBSY            ;WAIT
DCCB C9        669          RET
DCCC           670
DCCC           671 WNBSY:          ;WAIT ROUTINE (LOOP UNTIL DEVICE IS READY.)
DCCC C5        672          PUSH   BC
DCCD 06 20     673          LD     B,20H            ;*
DCCF           674 WNBSY0:
DCCF 10 FE     675          DJNZ   WNBSY0           ;*WAIT
DCD1           676
DCD1           677 WNBSY1:
DCD1 01 F8 0F  678          LD     BC,STR
DCD4 ED 78     679          IN     A,(C)
DCD6 4F        680          LD     C,A              ;SAVE   STATUS
DCD7 E6 81     681          AND    81H              ;CHECK
DCD9 20 F6     682          JR     NZ,WNBSY1        ;REPEAT
DCDB 79        683          LD     A,C              ;GET BACK STATUS
DCDC C1        684          POP    BC               ;BACK   BC
DCDD C9        685          RET
DCDE           686
DCDE           687 WAIT1:          ;WAIT ROUTINE 2
DCDE 3E 07     688          LD     A,7
DCE0           689 WAIT2
DCE0 3D        690          DEC    A
DCE1 20 FD     691          JR     NZ,WAIT2
DCE3 C9        692          RET
DCE4           693
DCE4           694 SETSCT:
DCE4 EB        695          EX     DE,HL
DCE5 7E        696          LD     A,(HL)           ;A=COM
DCE6 23        697          INC    HL
DCE7 56        698          LD     D,(HL)           ;D=SECTOR#
DCE8 23        699          INC    HL
DCE9 22 C4 DD  700          LD     (KDE),HL
DCEC 01 FA 0F  701          LD     BC,SCR           ;BC=SECTOR REG
DCEF ED 51     702          OUT    (C),D            ;SET SECTOR#
DCF1 F5        703          PUSH   AF               ;SAVE COM
DCF2 CD CC DC  704          CALL   WNBSY
DCF5 F1        705          POP    AF               ;GET BACK COM
DCF6 C9        706          RET
DCF7           707
DCF7           708 SETDMA:
DCF7 01 80 1F  709          LD     BC,$1F80
DCFA           710 STDM:
DCFA 04        711          INC    B
DCFB ED A3     712          OUTI
DCFD 15        713          DEC    D
DCFE 20 FA     714          JR     NZ,STDM
DD00 C9        715          RET
DD01           716
DD01           717 ER_TEXT:         EQU     80*22+3000H
DD01           718
DD01           719
DD01           720 ERROR:
DD01 16 95     721          LD     D,5+128+16       ;MEANS COLOR 5:CSIZE 2:CFLASH
 1
DD03 3A C3 D9  722          LD     A,(DRIVE)
DD06 B7        723          OR     A
DD07 20 05     724          JR     NZ,D1
DD09 21 72 DD  725          LD     HL,MES1
DD0C 18 03     726          JR     DO_PRT
DD0E           727 D1:
DD0E 21 87 DD  728          LD     HL,MES11
DD11           729 DO_PRT:
DD11 CD 3C DD  730          CALL   MES_PRINT
DD14 3E FF     731          LD     A,0FFH           ;ERROR MARK
DD16 18 3E     732          JR     P_EXIT
DD18           733 ERROR2:
DD18 16 95     734          LD     D,5+128+16
DD1A 21 9C DD  735          LD     HL,MES2
DD1D CD 3C DD  736          CALL   MES_PRINT
DD20 3E FF     737          LD     A,0FFH           ;ERROR MARK
DD22 18 32     738          JR     P_EXIT
DD24           739 ERROR3:
DD24 16 95     740          LD     D,5+128+16
DD26 21 AA DD  741          LD     HL,MES3
DD29 CD 3C DD  742          CALL   MES_PRINT
DD2C 3E FF     743          LD     A,0FFH           ;ERROR MARK
DD2E 18 26     744          JR     P_EXIT
DD30           745
DD30           746 ERROR4:
DD30 16 95     747          LD     D,5+128+16
DD32 21 B7 DD  748          LD     HL,MES4
DD35 CD 3C DD  749          CALL   MES_PRINT
DD38 3E FF     750          LD     A,0FFH           ;ERROR MARK
DD3A 18 1A     751          JR     P_EXIT
DD3C           752
DD3C           753 MES_PRINT:
DD3C           754          ;ENTRY
DD3C           755          ;      D=ATTRIBUTE
DD3C 4E        756          LD     C,(HL)
DD3D 23        757          INC    HL
DD3E 46        758          LD     B,(HL)
DD3F 23        759          INC    HL
DD40           760 TPL:
DD40 7E        761          LD     A,(HL)
DD41 B7        762          OR     A
DD42 C8        763          RET    Z
DD43 ED 79     764          OUT    (C),A
DD45 03        765          INC    BC
DD46 ED 79     766          OUT    (C),A
DD48           767
DD48 CB A0     768          RES    4,B
DD4A ED 51     769          OUT    (C),D
DD4C 0B        770          DEC    BC
DD4D ED 51     771          OUT    (C),D
DD4F CB E0     772          SET    4,B
DD51           773
DD51 03        774          INC    BC
DD52 03        775          INC    BC
DD53 23        776          INC    HL
DD54 18 EA     777          JR     TPL
DD56           778
DD56           779 P_EXIT:
DD56 32 09 D8  780          LD     (STATUS),A
DD59 11 B5 D9  781          LD     DE,RESTORE0
DD5C CD E1 DB  782          CALL   DIOCS
DD5F 11 BC D9  783          LD     DE,RESTORE1
DD62 CD E1 DB  784          CALL   DIOCS
DD65 18 04     785          JR     EXIT2:
DD67           786 EXIT:
DD67 AF        787          XOR    A
DD68 32 09 D8  788          LD     (STATUS),A
DD6B           789 EXIT2:
DD6B F3        790          DI
DD6C ED 7B E8  791          LD     SP,(SVSP)
DD6F DD
DD70 FB        792          EI
DD71 C9        793          RET
DD72           794
DD72           795 MES1:
DD72 F6 36     796          DW     ER_TEXT+22
DD74 52 45 41  797          DB     "READ ERROR DRIVE 0",0
DD77 44 20 45
DD7A 52 52 4F
DD7D 52 20 44
DD80 52 49 56
DD83 45 20 30
DD86 00
DD87           798 MES11:
DD87 F6 36     799          DW     ER_TEXT+22
DD89 52 45 41  800          DB     "READ ERROR DRIVE 1",0
DD8C 44 20 45
DD8F 52 52 4F
DD92 52 20 44
DD95 52 49 56
DD98 45 20 31
DD9B 00
DD9C           801 MES2:
DD9C 02 37     802          DW     ER_TEXT+34
DD9E 57 52 49  803          DB     "WRITE ERROR",0
DDA1 54 45 20
DDA4 45 52 52
DDA7 4F 52 00
DDAA           804 MES3:
DDAA 02 37     805          DW     ER_TEXT+34
DDAC 53 45 45  806          DB     "SEEK ERROR",0
DDAF 4B 20 45
DDB2 52 52 4F
DDB5 52 00
DDB7           807 MES4:
DDB7 00 37     808          DW     ER_TEXT+32
DDB9 4E 4F 54  809          DB     "NOT SAME",0
DDBC 20 53 41
DDBF 4D 45 00
DDC2           810 BUFAD:
DDC2 00 D0     811          DW     0D000H    ;読みこむアドレス
DDC4           812 KDE:
DDC4 00 00     813          DS     2
DDC6           814 KEEPCOM:
DDC6 00 00     815          DS     2
DDC8           816 BYTES: EQU     256
DDC8           817
DDC8           818 R_DMA:          ;15 Bytes
DDC8 83 7D FB  819   ·      DB     83H,7DH,0FBH,0FH
DDCB 0F
DDCC FF 00     820          DW     BYTES-1
DDCE 2C 10 80  821          DB     2CH,10H,80H,8DH
DDD1 8D
DDD2           822 RD_AD:
DDD2 00 C0     823          DB     00H,0C0H
DDD4 92 CF 87  824          DB     92H,0CFH,87H
DDD7           825
DDD7           826 W_DMA:          ; 17 Bytes
DDD7 83 79     827          DB     83H,79H
DDD9           828 WR_AD:
DDD9 00 C0     829          DB     00H,0C0H
DDDB FF 00     830          DW     BYTES-1
DDDD 14 28 80  831          DB     14H,28H,80H,8DH,0FBH,0FH,92H,0CFH,05H,0CFH,87H
DDE0 8D FB 0F
DDE3 92 CF 05
DDE6 CF 87
DDE8 00 00     832 SVSP:   DS     2
DDEA 00 00 00  833          DS     128
DDED 00 00 00
DDF0 00 00 00
DDF3 00 00 00
DDF6 00 00 00
DDF9 00 00 00
DDFC 00 00 00
DDFF 00 00 00
DE02 00 00 00
DE05 00 00 00
DE08 00 00 00
DE0B 00 00 00
DE0E 00 00 00
DE11 00 00 00
DE14 00 00 00
DE17 00 00 00
DE1A 00 00 00
DE1D 00 00 00
DE20 00 00 00
DE23 00 00 00
DE26 00 00 00
DE29 00 00 00
DE2C 00 00 00
DE2F 00 00 00
DE32 00 00 00
DE35 00 00 00
DE38 00 00 00
DE3B 00 00 00
DE3E 00 00 00
DE41 00 00 00
DE44 00 00 00
DE47 00 00 00
DE4A 00 00 00
DE4D 00 00 00
DE50 00 00 00
DE53 00 00 00
DE56 00 00 00
DE59 00 00 00
DE5C 00 00 00
DE5F 00 00 00
DE62 00 00 00
DE65 00 00 00
DE68 00 00
DE6A           834 USP:
DE69 [8669]
```

リスト5

```
10 WIDTH 80:CLEAR &HD000:PALET 4,7
20 LOADM "OBJ"
30 AD=&HD800:ST=AD+9
40 CSIZE2:LOCATE 6,0:PRINT#0,"---- EASY DISK UTILITY ---- By Z.N
50 CSIZE2:LOCATE 16,2:PRINT#0,"1. FORMAT DISK  ";:CREV1:CSIZE:PR
INT#0,"(DRIVE1)":CREV
60 CSIZE2:LOCATE 16,4:PRINT#0,"2. DISK COPY    ";:CREV1:CSIZE:PR
INT#0,"(DRIVE0 TO DRIVE1)":CREV
70 CSIZE2:LOCATE 16,6:PRINT#0,"3. DISK COMPARE ";:CREV1:CSIZE:PR
INT#0,"(DRIVE0 & DRIVE1)":CREV
80 CSIZE2:LOCATE 16,8:PRINT#0,"4. EXIT"
90 CSIZE2:LOCATE 16,10:PRINT#0,"   SELECT[1-4]:";:PRINTCHR$(26);
100 A$=INKEY$(1)
110 ON VAL(A$) GOTO "FMT","CPY","CMP","END":GOTO90
120 GOTO 90
130 LABEL"FMT"
140 GOSUB"RDY"
150 CALL AD+3
160 IF PEEK(ST) THEN END ELSE 50
170 LABEL"CPY"
180 GOSUB"RDY"
190 CALL AD+0
200 IF PEEK(ST) THEN END ELSE 50
210 LABEL"CMP"
220 GOSUB"RDY"
230 CALL AD+6
240 IF PEEK(ST) THEN END ELSE 50
250 LABEL"RDY"
260 PRINT#0,A$:CSIZE2:LOCATE 18,16:PRINT#0,"PUSH [S] TO START"
270 A$=INKEY$
280 IF A$="" GOTO270
290 IF INSTR("Ssト",A$) THEN 310
300 GOTO270
310 LOCATE 18,16:PRINTCHR$(26)
320 LOCATE 0,20:PRINTSTRING$(80,&HEF)
330 RETURN
340 LABEL"END"
350 END
```

micro computer 入門

周辺LSIを使いこなそう(2)

X68000のハードウェア操縦法

パソコンプログラミングの道を突き進んで行けば一度はぶつかる壁「ハードウェア」。ここでは，DMA，AD PCM，CRTCなどX68000のハードウェア資源を直接，効率よく制御する方法をリストを交えながら紹介します。

Kuwano Masahiko
桒野 雅彦

はじめに

プログラムを自分で組むようになってくると，そのうちどこかでハードウェアという壁にぶつかります。それまでは高級言語にせよ，アセンブラにせよ，メモリやレジスタ間でデータの受け渡しや演算を行い，ライブラリやIOCS (Input Output Control System)/DOSコールを重ねていけばよかったものが，突然わけのわからないビットパターンのデータ転送になり，なぜかそれで目的が達成される世界。これをやるときはこちらにこのデータを書き込んでから行ってください。という理由も教えない一方的な指示……。ここであきらめてしまう人が多いことも納得できなくはありません。

おまけに最近では強い味方がついています。そう，ソフトウェアの共通化という言葉です。どの機械の上でも同じソフトが走るようにしましょう，ハードを直接アクセスするのは御法度ですという言葉です。新興宗教にも似たこの動きでMSX，AXが生まれ，そして今度はOS/2も標準化しようという動きがちらほらしているようです。まあ，漢字が使用できて簡単な線画が描ければ十分だというビジネス用の機械などは，とっととAXと98に統一してしまえばよいのでしょうが，個人用となるとそうはいきません。

求められるハードウェアの基本性能がほぼ画一なビジネス用と違って，まったく何をするかわからない個人用というのは実にバラエティに富んでいます。ある人は40文字×25行のキャラクタ画面だけでも十分に画白くやっていけますし，またある人はX68000でもまだまだ貧弱だと思うでしょう。OSにしてもS-OSくらいのほうが，自由に暴れられてよいという考え方もあれば，UNIXなんて不自由なOSは嫌いだという意見もあります。

パーソナルコンピュータに対する見方がずっと幅広い個人ユーザー群にとっては，プログラムはやはりハードウェアの機能を生かし切ったものでありたいと思うのは当然でしょう。それを突き詰めていくと，CなどでANSIの規格案の範囲だけでプログラムを組むのではもの足りなくなって専用ライブラリを使うようになります。そうして，次のステップとしてアセンブラが登場してDOSコールだ，IOCS コールだとなっていくのは当たり前のことであると思います。となってくると，その先にあるのはやはりハードウェアであり，周辺LSIを直接コントロールする世界なのです。

今回は入門的な意味合いも含めてX68000の周辺LSIのいくつかをいじってみようと思います。X68000の専用LSIの構造などは，ページ数の都合もあってほとんど触れられませんでしたのでアスキーから出ているX68000テクニカルデータブックなどを参考にしてください。少々わかりにくいマニュアルですが，プログラムのサンプルと併せて眺めれば，理解しやすいと思います。サンプルはあくまで触れてみるといった程度なのであまり実用的とは言えませんが，それだけにパラメータの変更などは容易でしょう。いろいろ試してみてください。

DMAコントローラ

DMA (Direct Memory Access)というのは，CPUを介さずに直接，メモリにアクセスすることを言います。コンピュータの心臓は言うまでもなくCPUですが，ハードウェア的に見た場合，CPUというのはかなり低速なデバイスなのです。これは，現在のコンピュータがすべてプログラムをメモリから読み，解釈し，実行するというサイクルを繰り返していることが原因です。この方法のおかげでかなり複雑な処理でさえ，割と簡単なハードウェアで実現できるというのは評価に値します。しかし，単なるデータ転送などの場合でもプログラムを読み出し，解釈しというステップを踏むことになっているため，無駄が多くなります。

また，外部からのデータ転送要求に素早く応答するというのもかなり難しいことです。たとえば，メモリの読み書きに100ns，1命令の実行時間も100nsとしましょう。このとき，「外部から要求があったら1μs以内に応答しなさい，ただし要求は1秒に1回です」と言われたらどうでしょう。10ns (1nsは0.001μs) 以下の時間が相手であるハードウェア（デジタル回路）にとっては，1μs などというのはかなりのんびりした時間になるため余裕をみた設計ができるでしょうが，このようなことをCPUにやらせようとすると，コンピュータとしてはまったく使い物にならなくなります。

まず，1μs 以内に応答するにはどうしたらよいでしょう。割り込みを使うと，割り込みが入った時点で割り込みベクタを読み出したり，帰り先やレジスタ，フラグの値をスタックに書き込むなどの手間がかかるため，1μs 以内に応答するのは難しくなってきます。また，ほかの割り込みの処理に入ってしまうと，その割り込み処理が終わるまで次の割り込みは受け付けられませんから，応答すべき割り込み以外はすべてマスクしておかなくてはなりません。タイマもキーボードも何も使えなくなってしまいます。それだけやったとしても 1μs という のはかなり厳しい数字です。

それならと，ステータスをチェックしながらループするようなプログラムを作ってしまったらそれこそ最悪です。当然，割り込み禁止で走ることになるうえに，1秒に1回のイベント(事象)を待ってループしていなくてはならないのです。つまり，ほかの仕事はまったくできなくなってしまいます。よって，データの転送のような，単純な仕事はハードウェアにまかせてしまおうということで考えられたのがDMAという手法なのです。DMA要求の処理は次のようになります。

1) DMA要求が発生すると，まずCPUに対してバスの解放要求が行われます。
2) CPUへ解放要求がくると，現在実行中

のバスサイクル(メモリのリード/ライトなど)が終了した時点で，バスを解放し自分自身の処理は一時中断して，バス解放要求が解除されるのを待ちます。

3) DMA要求をした側は,バス解放要求が受け付けられたのを見て，データの転送を行います。

4) 転送が終了したら，解放要求を取り下げます。

5) CPUは要求が取り下げられたのを見て中断していた処理を再開します。

DMA動作のポイントはCPUに対するバスの解放要求です。Z80，68000などの汎用マイクロプロセッサには，外部からの信号によってCPUの動作を一時停止させ，メモリなどをアクセスするための信号（バス）をすべてドライブしなく（解放）したり，その状態から再度継続して動作を開始させたりする機能があります。DMAコントローラはCPUのこの機能を利用して，CPUにバスを解放させ，自分がCPUになりすまし，CPUと同じようなタイミングでメモリなどをアクセスします。

この一連の流れはプログラムの内容とは関係なく，純粋にハード的な動作で行われます。バスの解放なども，CPUの割り込み禁止などとはまったく関係なく実行されます。もちろん，DMAが動いている間だけCPUの動作が遅くなるのですが，よほど頻繁にDMA転送が行われない限り，DMAが動いていることを意識することはありません(囲み参照)。

DMAを使うことで，データ転送の間CPUが待ちぼうけをくらうこともなくなるうえ，外部からの転送要求に対する応答性も抜群によくなります。要求があってからデータ転送が開始されるまでの時間は最悪でもCPUのバスサイクル1回分です。今回の例なら100nsです。データを読むまでの時間を入れても200ns。余裕たっぷりです。

このように，高速のデータ転送を効率よく行おうということになると，DMAに頼らざるを得なくなってきます。ところが，このような仕掛けを1から作るとなると，CPUとのやりとりやCPUになりすますための回路などだけでも，けっこうな大きさになってしまいます。そこで，各プロセッサのメーカーは自社のプロセッサに直結してDMAを簡単に行えるようにするLSI，DMAコントローラを作っています。

X68000には4チャンネル分が1チップになったDMAコントローラ，63450を積んでおり，うちチャンネル0，1，3の3つのチャンネルはそれぞれフロッピーディスク，ハードディスク，音声合成に使われています。残るチャンネル2は空きになっていて，ユーザー側で使用しても構わないようです。

DMAコントローラを使ってみよう

それでは試しにDMAコントローラを使ってみることにしましょう。まず，63450の持っているレジスタ一覧を図1に示します。チャンネル0の動作に関係するレジスタは00E84000Hから00E84039Hまでで，チャンネル1，2，3にはそれぞれ40H，80H，C0Hを足したアドレスに同じ機能をもつレジスタが配置されています。たとえばチャンネル0のOCR0(オペレーション・コントロール・レジスタ)は00E84005H番地ですから，チャンネル1,2,3のOCRφはそれぞれ00E84045H，00E84085H，00E840C5H 番地にあることになります。

1チャンネル分だけでこれだけのレジスタを持っているところから見てもわかるように，かなり高機能なDMAコントローラです。あまりにも多機能であるため，とりあえず今回使ううえで必要なビットについてレジスタごとに説明しておきましょう。

●CSR

DMAコントローラのステータスが入りますが，とりあえず無視しておいて構いません

●CER

DMA転送でエラーが発生した場合,CSR0のERRビットが立って，ここにエラーステータスが入ります。まあ，見る必要はないでしょう

●DCR

・XRM

XRMビットはX68000ではチャンネル0，1，3には10または11をセットします。チャンネル2は00,10,11のいずれをセットしても(01はDMAコントローラ側で未定義になっている)かまいません。いずれのチャンネルでも10をセットするとCPUの動作が少し優先されるようになります。

・DTYP

ふつうはDMAによる転送の方式を指定するのですが，X68000では，ここは00（デュアルアドレスモード）に固定になっています。

・DPS

デバイス・ポート・サイズの略です。X68000でデータ転送の対象になるI/Oは8ビット幅ばかりですから，I/Oが相手ならここは0(8ビットポート)です。メモリ・メモリ間の転送をするときは1（16ビット）にしておきます。

・PCL

ここは00にしておきます。

●OCR

・DIR

データ転送方向を示します。メモリ・メモリ間転送の場合に限らず，DMA転送がデュアルアドレスモードで行われるとき，DMAコントローラはI/Oとメモリのアドレスを両方知らなくてはなりません。このため，I/O側用としてDAR（デバイス・アドレス・レジスタ)，メモリ側用にMAR(メモリ・アドレス・レジスタ）を持っています。DIRはこの両方のレジスタのどちらからどちらに転送するのかを決めるもので，0のときはMARの示すアドレスからDARの示すアドレスへの転送（I/Oへの書き込み)に，1のときはその反対（I/Oからのリード)になります。

・BTD

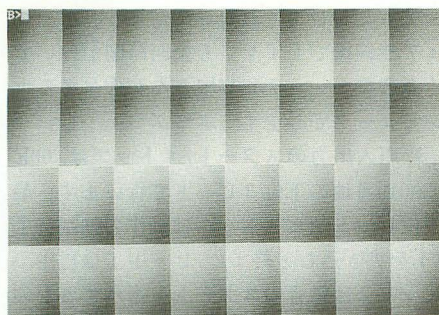
0に固定しておきます。

・SIZE

データ転送をバイト（8ビット）単位で行う（=00）か，ワード（16ビット）単位で行う（=01）か，ロングワード（32ビット）単位で行うか（=10）の区別をします。通常はDCRのDPSビットで指定したサイズに合わせておけばよいでしょう。メモリ・メモリ間転送の場合にはどれを選んでも構いません。このときDMA転送はSIZEで指示した分が1回の転送としてカウントされます。MTCに転送回数をセットするときに，このことに注意しておかないと，とんでもないところまでDMA転送されてしまいます。

・CHAIN

63450はメモリ上に転送を実行したいアドレスとカウント数の一覧表（転送情報テーブル)を置いておくと，それを自分で読み取って勝手に転送を次々と処理していく機能があります。この機能にも2種類あって，ひとつはアレイチェイニング，もうひとつはリンクアレイチェイニングと名づけられています。アレイチェイニングは転送情報テーブルが連続して配置される必要がありますが，リンクアレイチェイニングでは情報テーブルが分散して配置されても構わないという特徴があります。チェインテーブルの構造などについては福袋やCコンパイラに付属のプログラマーズマニュアル（IOCSコール$8B, $8C）の個所を参照してください。

チェインしない，普通のデータ転送では00，アレイチェイニングなら10，リンクアレイチェイニングなら11にします。

グラフィックをDMA転送する

・REQG

データ転送要求がどこからくるかを指定します。チャンネル0，1，3 (FDD, HDD, AD PCM)では10，チャンネル2は01にしておきます。

●SCR

・MAC, DAC

MAC(メモリ・アドレスレジスタ・カウント）はMAR，DAC（デバイス・アドレスレジスタ・カウント）はDARが，それぞれ転送のたびに増加/減少するか否かを指定するものです。00なら変化なし，01なら増加，10なら減少になります。

●CCR

・STR

DMA動作のスタート/ストップを指示します。1でスタートです。

・CNT, HLT, SAB

DMA転送の継続や中断などをするものですが，通常は0のままでよいでしょう。

・INT

割り込みをかけるか否かを決めるビットです。こちらでDMAをいじって，割り込みが入ってもHumanはうまいことやってくれるでしょう。1(割り込み許可)にします。

●MTC

転送したい回数をセットします。転送サイズに気をつけてセットしてください。たとえばワード単位のときにMTCを4にすると8バイトの転送になります。

●MAR

データ転送時のメモリ側のアドレスを入れます。

●DAR

I/O側のアドレスを入れます（メモリ・メモリ転送ならメモリのアドレスになります)。

●BTC

アレイチェイニング動作をさせるときの，アレイの数を入れます。

●BAR

アレイチェイニング動作をさせる場合，転送情報テーブルの先頭アドレスを入れます。

●NIV, EIV

割り込みベクタを入れます。これらはHumanがすでにセットしているので，いじらないようにします。

●MFC, DFC, BFC

68000はメモリをアクセスするとき，FC0, FC1，FC2という3本の信号線を使って，アクセスはスーパーバイザとしてアクセスしているのか，ユーザーとしてアクセスしているのか，またデータなのか，コード(プログラム）なのかなどを外部に知らせます。MFC，DFC，BFCは，それぞれDMAコントローラがMAR，DAR，BARのアドレスを出力するとき，FC0～2をどのような状態にするかを決めるものです。通常は101(スーパーバイザ，データ領域）としておけばよいでしょう。

●CPR

チャンネルのプライオリティを決めます。これもHumanがすでにセットしてくれているはずなので，いじらなくてよいでしょう。

サンプルプログラム(リスト1)

サンプルは簡単にということで，アレイチェインも使わない単純なメモリ・メモリ

図1　63450のレジスタ構成

Ch No.	レジスタアドレス	D07 (D15)	D06 (D14)	D05 (D13)	D04 (D12)	D03 (D11)	D02 (D10)	D01 (D09)	D00 (D08)	備考
0Ch	E84000H	COC	BTC	NDT	ERR	ACT	DIT	PCT	PCS	CSR0（チャンネルステータスレジスタ）
	E84001H	0	0	0	←ERROR CODE→					CER0（チャンネルエラーレジスタ）
	E84004H	←XRM→		←DTYP→		DPS	0	←PCL→		DCR0（デバイスコントロールレジスタ）
	E84005H	DIR	BTD	←SIZE→		←CHAIN→		←REQG→		OCR0（オペレーションコントロールレジスタ）
	E84006H	0	0	0	0	←MAC→		←DAC→		SCR0（シーケンスコントロールレジスタ）
	E84007H	STR	CNT	HLT	SAB	INT	0	0	0	CCR0（チャンネルコントロールレジスタ）
	E8400AH	←								MTC0（メモリトランスファカウンタ）
	E8400BH								→	
	E8400CH	←								MAR0（メモリアドレスレジスタ）(H)
	E8400DH								→	
	E8400EH	←								MAR0（メモリアドレスレジスタ）(L)
	E8400FH								→	
	E84014H	←								DAR0（デバイスアドレスレジスタ）(H)
	E84015H								→	
	E84016H	←								DAR0（デバイスアドレスレジスタ）(L)
	E84017H								→	
	E8401AH	←								BTC0（ベーストランスファカウンタ）
	E8401BH								→	
	E8401CH	←								BAR0（ベースアドレスレジスタ）(H)
	E8401DH								→	
	E8401EH	←								BAR0（ベースアドレスレジスタ）(L)
	E8401FH								→	
	E84025H	←							→	NIV0（ノーマルインタラプトベクタ）
	E84027H	←							→	EIV0（エラーインタラプトベクタ）
	E84029H	0	0	0	0	0	FC2	FC1	FC0	MFC0（メモリファンクションコードレジスタ）
	E8402DH	0	0	0	0	0	0	←CP→		CPR0（チャンネルプライオリティレジスタ）
	E84031H	0	0	0	0	0	FC2	FC1	FC0	DFC0（デバイスファンクションコードレジスタ）
	E84039H	0	0	0	0	0	FC2	FC1	FC0	BFC0（ベースファンクションコードレジスタ）
	（注1）									
3Ch	E840FFH	0	0	0	0	←BT→		←BR→		GCR（ジェネラルコントロールレジスタ）

注1：チャンネルNo.1は，チャンネルNo.0のアドレス＋40H
チャンネルNo.2　〃　＋80H
チャンネルNo.3　〃　＋C0H
※CER（チャンネルエラーレジスタ）のみREAD ONLY，その他のレジスタは，READ/WRITE可

転送をグラフィックVRAM 同士の間で行ってみました。CPUでインクリメントしたデータを画面の上4分の1のところにセットしたあと，画面半分までDMAで転送した後，画面下半分に今度は下から転送しています。VRAMが相手なので，どうしてもウェイトが入ってしまい，DMAコントローラの能力全開バリバリギンギンとはいきませんが，それでもまあまあ速いとは言えるのではないでしょうか。

AD PCMまわりを少し見てみましょう

本来はFM音源では難しい音声合成を目的としていたのに，なぜかビープ音になり，ゲームミュージックではバスドラなどに使われているAD PCM(MSM6258)を動かしてみましょう。

AD PCMは外部から入力されたアナログデータを定周期で取り込み，前回の入力値との差分を4ビットのデータにしています。再生の場合はこの逆で次々に差分を取ったデータを与えると元の波形を再現するのです（録音は最大振幅が8ビット分まで，再生は10ビット分になっています)。4ビットのデータのうち最上位ビットが電圧が増える方向なのか，減る方向なのかを決め，下位3ビットで変化の絶対値を表しています。

AD PCMのデータは電圧絶対値ではなく，前回との差分を取っていますので，加算方向や減算方向のデータばかり送っていると飽和してしまって音にならなくなります。また，サンプリング周波数も最大で，15.6kHzですから，再生できるのはこの半分の7.8kHzまでです。

サンプリングが44kHzで，16ビットの絶対値であるCDなどと比べるとだいぶ見劣りがしますが，「音声合成」が目的だったのですから比べるのは酷というものでしょう。ちなみにアマチュア無線では振幅変調（AM，SSB)で音声を送るときの帯域幅は3kHzになっています。この値は男性の声を基準にしたようで，女性の場合にはカットされる成分が多くなってきて，ちょっと厳しいところです。7.8kHz ならまあ十分な値であると言えるでしょう。

AD PCM，およびAD PCMに関係するポートは図2のとおりです。これらのうち00E9A005H番地はジョイスティックポートに使っている8255のポートCの下位4ビット（PC0～3)，また00E92003H番地は FM音源ICの出力ポートで，それぞれパンポット制御やサンプリング周波数の切り替えに使われています。さて，周辺回路はというと，これは図3のようになっています。音声出力のほうは，この回路図の右端からさらにYM2131 (FM音源）とミキシングされてコネクタに出ています。

パンポット機能はAD PCMの出力を強引に0に引きずり落とすか否かによって行い，音を出す側を決めるのではなく，音を消す側を決めるようにしていることがわかります。また，入出力とも簡単なCRフィルタが入っています。極端に低い周波数によってAD PCMのデータが揺さぶられたり，出力にPCM特有の高い周波数成分のノイズのようなものが出ていかないようにしています。このため，たとえば直流電圧の測定などには使えません。入力コネクタがステレオジャックであるというのもなかなかのムードです。

AD PCMを使ってみる

AD PCMへのデータ転送はDMAが行い DMA コントローラのチャンネル3が AD PCM専用に割り当てられています。サンプリングレートは8255のPC2，3によるMSM6258自身が持つ選択機能と，MSM6258 自体の動作クロックをYM2151のポート（CT0）で切り替えることで行います(CT0が0だと8MHz，1だと4MHz)。クロックが 4MHzのときは3.9kHz (PC3,2が00), 5.2kHz (同01)，7.8kHz (同10）が選択でき，クロックを8MHzにすると2倍になります。

パンポットはPC0とPC1を使っていることは先ほども調べたとおりです。PC0だと右チャンネル，PC1だと左チャンネルの出力がカットされます。参考文献のテクニカルマニュアルでは両チャンネルCUTと両チャンネルOUTがさかさまになっていますので，もし手元にあったら直しておいたほうがよいでしょう。最初，マニュアルどおりにやって音が出ないためにずいぶん悩んで，回路図（元祖タイプのものがI/O誌の1987年7月号に載っています）を眺めて初めてマニュアルの間違いとわかりました。やっぱり回路図は貴重な資料です（なんでマニュアルに付けてくれないのだろう。アマチュア無線機でもTVでも全回路図が付いてくるのに……。テレビ屋さんが作ってるコンピュータでしょ！　ぶつぶつ……)。

音声入出力のスタート/ストップは 00E92001H番地の下位3ビットで行います。最下位ビット（ビット0）は動作の強制終了（1を書き込むと終了)，ビット1は再生，ビット2は録音のスタートを指示するもので，それぞれ1を書き込むとスタートします。ビット1とビット2を同時に1にするようなことはしないようにしてください。

あとはDMAの設定です。データの入出力ポートは00E92003H番地です。DMAのモードは外部転送要求，サイクルスチール，デュアルアドレスモードで行います。

サンプルプログラム(リスト2)

サンプルはグラフィック VRAM をデータの転送相手として選んでみました。DMAコントローラの動作モードは簡単に単一データブロック転送（MTCサイズ分だけ転送）ですませています。アレイチェイニング機能を使えば画面一杯に録音/再生データを表示させることもできますが，それは宿題ということにしましょう。

CRTコントローラ

X68000の画面出力関係はかなり強力であるうえ，ほとんどの回路はシャープオリジナルのLSI (CRTコントローラ，ビデオコントローラ，スプライトコントローラ）であり，そのマニュアルなどは当然のことながら市販されていません。これらのコントローラの基本的な考え方は，次のようになっているようです。

CRTコントローラがディスプレイのための同期信号やメモリ読み出しのタイミングの作成を行います。このタイミングに合わせてスプライトコントローラがスプライトパターンを出力し，VRAM から出てきたデータと共にビデオコントローラに送られ

図2　ADPCM関連レジスタ構成

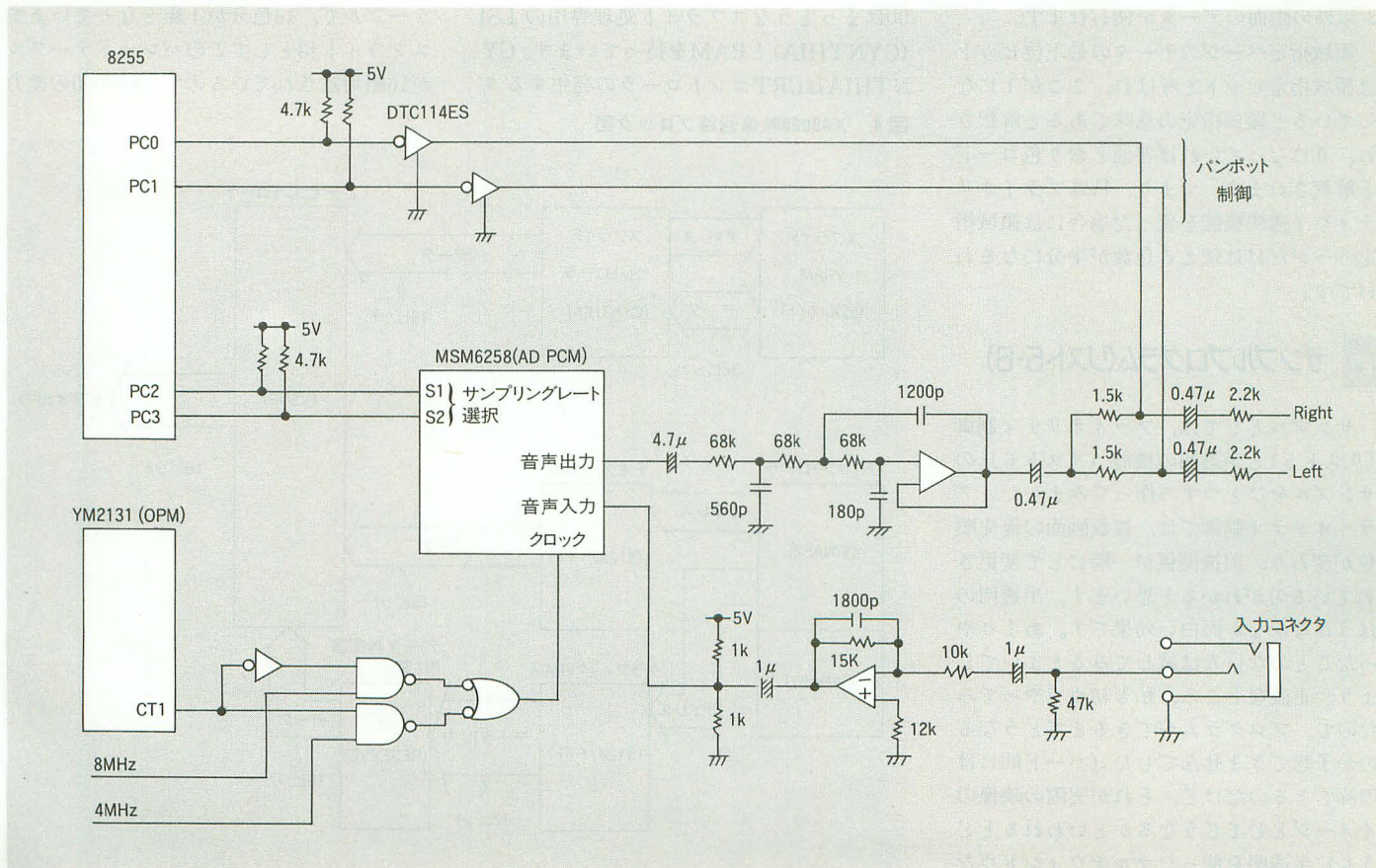
レジスタアドレス		D7	D6	D5	D4	D3	D2	D1	D0	備　考
E92001H	READ	REC/PLAY	1	0	0	0	0	0	0	AD PCMステータス
	WRITE	0	0	0	0	0	REC ST	PLAY ST	SP	AD PCMコマンド
E92003H	READ	B3	B2	B1	B0	B3	B2	B1	B0	入力データ
	WRITE									出力データ
E9A005H	WRITE	×	×	クロック切り替え(4MHz/8MHz)		Sampling RATE		PCMPAN		AD PCM出力/サンプリング周波数切り替え

ます。ビデオコントローラはこれらのデータの間でプライオリティや半透明の処理を行います。

CRTCをいじってみよう

CRTコントローラ(VINAS1/VINAS2)は，VRAMの読み出しタイミングやディスプレイへの同期信号の発生などのコントロールを行っています。ハードウェアスクロールは，VRAMの読み出しを始めるアドレスを変更することで，水平周波数の変更はディスプレイへの同期信号の変更で行われています(以下，コントローラLSIの名前は初代X68000用の回路を規準とする。ACE以降は名前が違う)。

ディスプレイ側へのタイミングの変更は，ディスプレイ側が無段階に追尾してくれるタイプの場合にはいろいろと面白い事ができそうなのですが，CZ-600DEはリレー切り替えなので，変更しても同期が取れなくなるだけで面白くありません。

ここではVRAMの読み出しタイミングの変更によってグラフィック画面が中央から湧き出してくるように見えるプログラムと，テキスト画面のハードウェアスクロールの2題をお届けしましょう。

湧き出るグラフィック(リスト3)

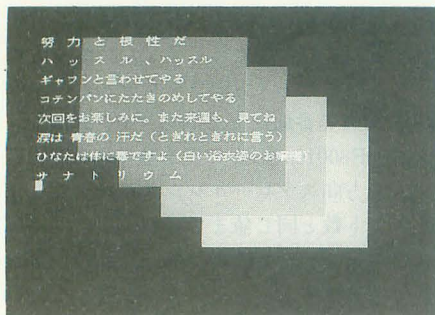
ひとつ目は湧き出るグラフィック。これは，同期信号を発生してからどれだけ後にVRAMの読み出しを始めるか（表示開始位置）を決めているレジスタと，どこから先はブランクにしてしまうか（表示終了位置）を決めているレジスタを操作しています。表示開始位置，終了位置は水平方向，垂直方向を独立に決定できるので，まず中央付近を表示開始位置と終了位置にしておき，あとは開始位置を左上のほう(水平，垂直とも減少方向)に，表示終了位置を右下(水平，垂直とも増加方向)に変更しているだけです。CRTCに限らず，ビデオ関係のLSIのレジスタの書き込みはMFPのGPIP4ビット（00E88001H番地のビット4）が0の間に行わないと気が狂ってしまうらしく，画面が妙なぐあいになる場合が見られました。

テキスト画面のスクロール

テキスト画面をスクロールさせるというのは，大方の場合，単なるいやがらせにしかなりません。これまであまり試す機会がありませんでしたので，ものは試しでちょっとやってみることにしました。やり方はとても簡単で，X方向スクロールレジスタ(00E80014H番地)とY方向スクロールレジスタ（00E80016H番地）に値を放り込むだけです。X方向スクロールレジスタは0から水平ドット数と同じ値までの数値を，Y方向スクロールレジスタには0～1023の数値を書き込めばずるっと，スクロールしてくれます。サンプルではスクロールさしあとに再び元の位置までスクロールし直しています（リスト4）。

ビデオコントローラ

ビデオコントローラ（VSOP：ビショップ）は，VRAMから出てきたデータの管理を引き受けています。回路図を眺めるとテキスト画面16ビット単位でそのままVSOPに送られています。グラフィックVRAMのデータは，32ビット単位で読み出されたあと（この設計もなかなか奥が深い）画面モードに応じてRESERVEと名づけられたLSIで16ビットのデータに再配列されてVSOPに送られています。またスプライトコントローラ（CYNTHIA：シンシア）が処理したスプライトデータは16ビットのデ

図3　AD PCM周辺回路図

プライオリティの制御

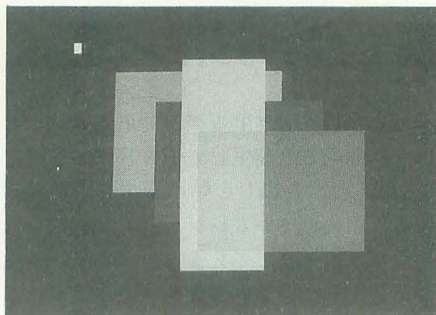
半透明機能

ータになり，VSOPに送られます（図4）。VSOPはこれらのデータを使って画面の表示ON/OFFや画面間のプライオリティの処理，特殊プライオリティ機能，半透明機能の実現などを行っています。

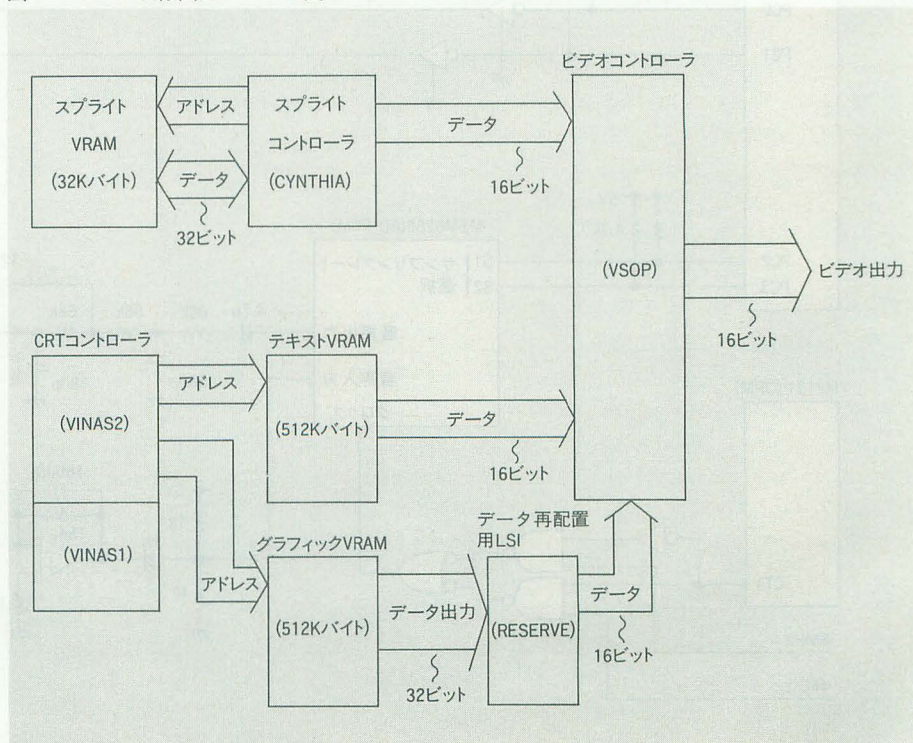
特殊プライオリティは，グラフィック画面の中でいちばんプライオリティの高い画面で指定した領域だけが部分的にテキストよりも高いプライオリティになる機能です。テキスト画面が，グラフィック画面の間に挟まれたような状態になります。半透明機能はグラフィック画面の中でいちばんプライオリティの高い画面（領域指定ページ）と，テレビ画面，テキスト画面，2番目にプライオリティの高いグラフィック画面のいずれかひとつのデータとの平均値を出力するものです。輝度ビットは領域指定ページ以外の画面のデータが使われます。

領域指定ページのデータの最下位ビットは領域指定ビットと呼ばれ，ここが1になっていると領域指定の意味であると解釈され，0になっていれば普通どおり色コードと解釈されます。つまり，特殊プライオリティや半透明機能を使った場合には領域指定ページだけは使える色数が半分になるわけです。

サンプルプログラム(リスト5・6)

サンプルとしては，プライオリティ制御（リスト5）と半透明の機能（リスト6）のサンプルをひとつずつ作ってみました。プライオリティ制御では，複数画面の優先順位が変わり，前後関係が一瞬にして変更されているのがわかると思います。半透明のほうはなかなか面白い効果です。あまりやったことのない人は試してみるとよいでしょう。正直なところ，私も初めてやってみたので，プログラムができるまでどうなるのか予想できませんでした（ハード的には理解できるのだけど，それが実際の映像のイメージとしてどうなるかといわれるとどうも)。半透明を使ったマルチウィンドウなんていうのも面白いかもしれません。

スプライト

スプライトについては，もう改めて説明するまでもないくらいでしょう。自分の定義したパターンを1ドット単位で好きなところに配置できるこの機能は，これまでゲームを作るときにどうしても面倒で，時間もかかる作業でしたが，キャラクタと背景，キャラクタ同士の重なりなどを自動的に処理してくれることからBASICでもアクションゲームが作れる，とずいぶん騒がれたものです。

TOWNSの場合には，定義されたパターンをグラフィックVRAMにDMA転送する，疑似スプライトを乗せたようですが，X68000はまっとうなスプライト処理専用のLSI (CYNTHIA) とRAMを持っています。CYNTHIAはCRTコントローラの発生するタイミングに従ってスプライトデータを出力しており，これがディスプレイコントローラ(VSOP)によってグラフィック画面やテキスト画面と合成されます。

映像信号を合成しているだけですから，いくら重なっても下になった部分のデータに気を使う必要はありません。

スプライトを動かしてみよう

スプライト動作はVINAS1/VINAS2 (CRTコントローラ)，CYNTHIA，VSOPの4者の協調作業です。画面の初期化などからやっていては，ごちゃごちゃするだけですから，ここではスプライトの定義に絞っておきましょう。

スプライトレジスタのマップを図5に示します。スプライトスクロールレジスタは128個のスプライトのそれぞれの表示位置，水平・垂直反転の有無，パレット番号，スプライトパターンの番号，プライオリティが登録されます。

パレット番号は16組あるパレットテーブルのどれを使うかを決めるものです。X68000のスプライトはひとつのパターンで65536色のなかの16色を使えるようになっています。この時0～15の色コードが65535色のどれに当たるかを決めているのがパレットテーブルで，16色分が1組となっています。スプライト用としてこのパレットテーブルが16組用意されているので，X68000の能力

図4　X68000映像回路ブロック図

としてはスプライトとテキストで16×16＝256色まで同時に出力できることになります。これらはBASICのサンプルを兼ねてオマケで付いてきたスプライト定義ツールなどでもお馴染みでしょう。さて，スプライトスクロールレジスタにはパターンの番号は書いてありますが，パターンの形そのものは定義していません。パターンデータ自体はPCGテーブルがあります。スプライトVRAMの32Kバイトがこのために割り当てられています。VRAMは画面の表示動作についていく必要からアクセスタイムが45nsという，高速のスタティックRAMを使用しています。

テクニカルマニュアルでもあまりはっきりと言っていませんが，スプライトスクロールレジスタのほうはCYNTHIA内部にあります。CYNTHIAはスクロールレジスタで定義されたスプライトパターンの位置情報を見て，どのパターンのデータを引っ張り出せばよいかを知り，スプライトVRAMをアクセスしてパターンを取ってくるわけです。PCGのデータ配置はBASICのスプライト定義のようにはいかず，少々変則的です。16×16ドットのスプライトを縦横に4分割し，それぞれの左上の色コード（4ビット）ずつを合わせて16ビットデータとした形になっています（左上のデータが最上位，続いて左下，右上，右下の順)。次の16ビットデータはそれぞれ，いま取り出したドットの右隣のデータを集めたものになっています。

図5-1　スプライトのレジスタ構成

名称		レジスタアドレス	D15	D14	D13	D12	D11	D10	D09	D08	D07	D06	D05	D04	D03	D02	D01	D00	備考
スプライトスクロールレジスタ	SP 0	EB0000	0	0	0	0	0	0	←X座標									→	1個のスプライトについて左の4ワードのレジスタが割り当てられる。
		EB0002	0	0	0	0	0	0	←Y座標									→	
		EB0004	VR	HR	0	0	←COLOR(SP)			→	←SP CODE							→	
		EB0006	0	0	0	0	0	0	0	0	0	0	0	0	0	0 ※	←PRW	→	VR：V反転（スプライト） HR：H反転（スプライト）
	⋮	⋮							⋮										
	SP 127	EB03F8	0	0	0	0	0	0	←X座標									→	
		EB03FA	0	0	0	0	0	0	←Y座標									→	
		EB03FC	VR	HR	0	0	←COLOR(SP)			→	←SP CODE							→	
		EB03FE	0	0	0	0	0	0	0	0	0	0	0	0	0	0 ※	←PRW	→	
バックグラウンドスクロールレジスタ	BG0	EB0800	0	0	0	0	0	0	←X座標									→	
		EB0802	0	0	0	0	0	0	←Y座標									→	
	BG1	EB0804	0	0	0	0	0	0	←X座標									→	
		EB0806	0	0	0	0	0	0	←Y座標									→	
	BGコントロール	EB0808	0	0	0	0	0	0 ※	DISP/CPU	0	0	0	0 ←TEXTSEL(BG1)	→	BG1 ON/OFF	0 ←TEXTSEL(BG0)	→	BG0	
画面モードレジスタ		EB080A	0	0	0	0	0	0	0	0	←H total							→	水平トータル
		EB080C	0	0	0	0	0	0	0	0	0	0	←H disp					→	水平表示開始位置
		EB080E	0	0	0	0	0	0	0	0	←V disp							→	垂直表示開始位置
		EB0810	0	0	0	0	0	0	0	0	0	0	0	L/H freq	0 ←V Res.	→	0 ←H Res.	→	解像度

［スプライト，バックグラウンドスクロールレジスタ］（※は拡張用）（全レジスタREAD/WRITE可）

サンプルプログラム(リスト7)

ものは試しということで，アセンブラでスプライトをいじってみました。最初，定義したスプライトを何も考えずに動かしたら速すぎて冗談にもならなかったので，ここではけっこうな長さのディレイをかませています。

パターンやパレットのデータはご想像どおり，おまけのスプライト定義ツールが作ったBASICのテキストから行番号などを削ってアセンブラのソースの中にハメ込んだものです。パターンの定義は，このデータからPCGのデータに変換しながらセットしています。かなり泥臭い方法ですが，なんとかうまくいっています。高速化を目指すなら，あらかじめPCGテーブルにセットするデータフォーマットに直しておくほうがよいのでしょうが，テストのときにいちいちパターンを変更するたびにフォーマット変換をしているのがばかばかしくなったので，このようにしています。

言い忘れましたが，パレット番号0はテキスト画面のパレットとしても使われていますので，あまりいじりたくないところです。サンプルでは1番のパレットを使っています。

おわりに

私はそこまでやる気はないよ，と思っていてもハードウェアについて知ることはソフトウェアを組む上でも少なからず利点があるはずです。自分の行っている操作がハード的にどのように実現されているかを知れば，ハードウェアがより能率よく動けるようなプログラムの組み方というものもできるでしょうし，原理的に同時に行ってはいけない動作というものも理解できるでしょう。ある操作がIOCSレベルでもサポートされていないようなときでも，それが物理的に不可能なのか，それとも自分で少しハードウェアにアクセスしてやれば可能なのかの区別もつけられます。

そしてなによりも大きいのは，これまでブラックボックスになっていたハードウェアが少しでも見渡せるようになることで，ずっと身近な感覚でコンピュータを使うことができるということです。せっかく，これだけの投資をして手に入れたコンピュータです。しゃぶり尽くさなければもったいないではないですか。

ではまたお会いしましょう。

P.S.いま，ちょっとしたものを作っています（来月号のナニとはまた別です)。うまくいけば来年早々にでも紹介できると思います。お楽しみに。

図5-2 スプライトのレジスタ構成

名称		レジスタアドレス	D15	D14	D13	D12	D11	D10	D09	D08	D07	D06	D05	D04	D03	D02	D01	D00	備考
SPコード0	0	EB8000	I	G	R	B	I	G	R	B	I	G	R	B	I	G	R	B	・各SPコードに対するPCG配置
			←0→				←1→				←2→				←3→				
		EB8002	I	G	R	B	I	G	R	B	I	G	R	B	I	G	R	B	0 2
			←4→				←5→				←6→				←7→				(以下同) 16Bit
		⋮									⋮								1 3
		EB801C	I	G	R	B	I	G	R	B	I	G	R	B	I	G	R	B	
			←56→				←57→				←58→				←59→				16Bit
		EB801E	I	G	R	B	I	G	R	B	I	G	R	B	I	G	R	B	
			←60→				←61→				←62→				←63→				・水平512ドットモード
	1	EB8020	I	G	R	B	I	G	R	B	I	G	R	B	I	G	R	B	SPコードサイズ＝上図(0123)
			←64→				←65→				←66→				←67→				＝BGコードサイズ ＝16×16ドット
		⋮									⋮								・水平256ドットモード
		EB803E	I	G	R	B	I	G	R	B	I	G	R	B	I	G	R	B	
			←124→				←125→				←126→				←127→				SPコードサイズ＝上図(0123) ＝16×16ドット
	2	EB8040	I	G	R	B	I	G	R	B	I	G	R	B	I	G	R	B	BGコードサイズ＝上図(0)OR
		⋮	←128→				←129→				←130→				←131→				上図(1)OR
											⋮								上図(2)OR
		EB805E	I	G	R	B	I	G	R	B	I	G	R	B	I	G	R	B	上図(3)
			←188→				←189→				←190→				←191→				＝8×8ドット
	3	EB8060	I	G	R	B	I	G	R	B	I	G	R	B	I	G	R	B	
		⋮	←192→				←193→				←194→				←195→				
											⋮								
		EB807E	I	G	R	B	I	G	R	B	I	G	R	B	I	G	R	B	
			←252→				←253→				←254→				←255→				
SPコード1		EB8080	I	G	R	B	I	G	R	B	I	G	R	B	I	G	R	B	
		⋮	←0→				←1→				←2→				←3→				
											⋮								
			←252→				←253→				←254→				←255→				
		EB80FE	I	G	R	B	I	G	R	B	I	G	R	B	I	G	R	B	
SPコード2		EB8100	I	G	R	B	I	G	R	B	I	G	R	B	I	G	R	B	
		⋮	←0→				←1→				←2→				←3→				
											⋮								
			←252→				←253→				←254→				←255→				
		EB817E	I	G	R	B	I	G	R	B	I	G	R	B	I	G	R	B	
⋮		⋮									⋮								
SPコード127		EBBF80	I	G	R	B	I	G	R	B	I	G	R	B	I	G	R	B	
		⋮	←0→				←1→				←2→				←3→				
											⋮								
			←252→				←253→				←254→				←255→				
		EBBFFE	I	G	R	B	I	G	R	B	I	G	R	B	I	G	R	B	

［PCGエリア］（全レジスタREAD/WRITE可）

IBM-PC/ATの速度調節

MS-DOS（PC-DOS）機では，8086/8088 から最近の80386を使った機械まで，環境としてはまったく同一で，ただ処理速度が違うだけということになっています。ところがこの唯一の違いである速度が速くなったために困ったことが起こる場合が出てきました。ソフトウェアで適当に時間稼ぎのループをしているようなプログラムはCPUの処理速度が上がったために待ち時間が少なくなりすぎて，妙に忙しい動きになったり，本来稼がれるはずの時間が少なくなるためにタイミングが狂ってしまったりしてしまうのです。ゲームなどは被害者の最たるもので，処理速度２倍と言えば聞こえはよいのですが，たとえばスーパーハングオンやアフターバーナーが倍速で走ったりしたら，これは狂気の世界でしょう（驚喜する人もいるでしょうが）。

この解決として，PC-9801やPC-286/386などでは，CPUのクロックをスイッチで切り替え，動作速度を落とせるようにしていますが，IBMはPC/ATで別の方法をとりました。なんと，DMAコントローラを利用するのです。IBM-PCでは（PC-9801も同じですが）D-RAMのリフレッシュを行うための専用のD-RAMコントローラを使わず，タイマによって周期的にDMAコントローラにスタートを走らせる方法をとっています。

このDMAコントローラとCPUの間にちょっと細工をして，DMAコントローラがバスの解放要求を取り下げても，しばらくの間，CPUにそのことを伝えないようにしてしまうのです。この時間を調節することで，見かけの処理速度を下げてしまおうという魂胆なのです。CPUはこの遅らされた要求取り下げから，次のDMAの周期までの間だけ動けることになります。もし，伝えるのを遅らせる時間がDMAの周期以上になってしまえば，CPUはまったく動けなくなります。

この方式のメリットは速度調節がソフトウェアによって自在に変えられることです。実際，IBM-PC/ATではCTRL，ALT，Pageup/Pagedownの３つのキーを同時に押すことで，処理速度をいつでも上下できるようになっています。

リスト１ DMA動作テストプログラム（DMA.S）

```
 1: *------------------------------------
 2: *- ＤＭＡ動作テストプログラム        -
 3: *------------------------------------
 4: *
 5: *IOCSｺｰﾙ
 6: *
 7: _G_CLR_ON  equu              $90
 8: _CRTMOD    equ               $10
 9: _B_KEYINP  equ               $00
10: *
11: *DOSｺｰﾙ
12: *
13: _EXIT      equ               $ff00
14: _SUPER     equ               $ff20
15: *
16: *etc.
17: *
18: iocs       equ               $0f
19: G_VRAM     equ               $00c00000
20:            .data
21: SP_BUF:    dc.l              0
22:            .text
23:            clr.l             -(sp)
24:            dc.w              _SUPER
25:            addq.l            #4,sp
26:            move.l            d0,SP_BUF
27:            bsr               crtinit
28:            bsr               crtpatset
29:            bsr               keywait
30:            bsr               dmaset
31:            move.l            SP_BUF,-(sp)
32:            dc.w              _SUPER
33:            addq.l            #4,sp
34:            dc.w              _EXIT
35: *
36: * ｸﾞﾗﾌｨｯｸ画面を初期化(512×512、65535色)
37: * した後、ｸﾞﾗﾌｨｯｸ表示をｲﾈｰﾌﾞﾙします
38: *
39: crtinit:
40:            move.w            #12,d1
41:            moveq.l           #_CRTMOD,d0
42:            trap              #iocs
43:            moveq.l           #_G_CLR_ON,d0
44:            trap              #iocs
45:            rts
46: *
47: * ｸﾞﾗﾌｨｯｸVRAMにﾊﾟﾀｰﾝを書き込みます
48: *
49: crtpatset:
50:            movea.l           #G_VRAM,a0
```

```
 51:            moveq.l       #$0,d0
 52:            move.w        #$ffff,d1
 53:    patwrite_loop:
 54:            move.w        d0,(a0)+
 55:            addq.l        #$1,d0
 56:            dbra          d1,patwrite_loop
 57:            rts
 58: *
 59: * ｷｰ入力があるまで待ちます
 60: *
 61: keywait:
 62:            moveq.l       #_B_KEYINP,d0
 63:            trap          #iocs
 64:            tst.l         d0
 65:            beq           keywait
 66:            rts
 67: *
 68: * DMA転送実行
 69: *
 70: CSR2            equ           $00e84080
 71: CER2            equ           $00e84081
 72: DCR2            equ           $00e84084
 73: OCR2            equ           $00e84085
 74: SCR2            equ           $00e84086
 75: CCR2            equ           $00e84087
 76: MTC2            equ           $00e8408a
 77: MAR2            equ           $00e8408c
 78: DAR2            equ           $00e84094
 79: BTC2            equ           $00e8409a
 80: BAR2            equ           $00e8409c
 81: NIV2            equ           $00e840a5
 82: EIV2            equ           $00e840a7
 83: MFC2            equ           $00e840a9
 84: CPR2            equ           $00e840ad
 85: DFC2            equ           $00e840b1
 86: BFC2            equ           $00e840b9
 87: FC_USER_DATA        equ       1
 88: FC_USER_PROG        equ       2
 89: FC_SUPER_DATA       equ       5
 90: FC_SUPER_PROG       equ       6
 91: FC_INTA             equ       7
 92:
 93: dmaset:
 94:    move.b        #$18,CCR2        * 動作ｽﾀｰﾄしない
 95:    move.b        #$08,DCR2        * あまり意味はない
 96:    move.b        #$21,OCR2        * 最大転送速度32ﾋﾞｯﾄ転送
 97:    move.b        #FC_SUPER_DATA,MFC2
 98:    move.b        #FC_SUPER_DATA,DFC2
 99:    move.b        #$05,SCR2        * 両方とも増加
100:    move.l        #G_VRAM,MAR2     * ｿｰｽｱﾄﾞﾚｽ
101:    move.l        #G_VRAM+$00020000,DAR2 * ﾃﾞｨｽﾃｨﾈｰｼｮﾝｱﾄﾞﾚｽ
102:    move.w        #$8000,MTC2
103:    move.b        #$88,CCR2        * 動作ｽﾀｰﾄ
104:    bsr           wait_dma_end
105:    move.b        #$06,SCR2        * ｿｰｽは増加、相手は減少
106:    move.l        #G_VRAM,MAR2     * ｿｰｽｱﾄﾞﾚｽ
107:    move.l        #G_VRAM+$0007fffc,DAR2  * 最終ｱﾄﾞﾚｽ
108:    move.w        #$8000,MTC2
109:    move.b        #$88,CCR2        * 動作再ｽﾀｰﾄ
110:    bsr           wait_dma_end
111:    move.w        #$8000,MTC2
112:    move.b        #$88,CCR2        * 動作再ｽﾀｰﾄ
113:    bsr           wait_dma_end
114:    rts
115: *
116: * DMA転送が終了するのを待ちます
117: *
118: wait_dma_end:
119:    tst.w         MTC2
120:    bne           wait_dma_end
121:    rts
122:    .end
```

リスト2 ADPCM動作テスト(ADPCM.S)

```
  1: *-----------------------------------
  2: *- ＡＤＰＣＭ動作テスト                  -
  3: *- キーを叩くと録音開始し、              -
  4: *- 再度叩くと再生します                  -
  5: *-----------------------------------
  6: *
  7: *IOCSｺｰﾙ
  8: *
  9: _G_CLR_ON  equ          $90
 10: _CRTMOD    equ          $10
 11: _B_KEYINP  equ          $00
 12: *
 13: *DOSｺｰﾙ
 14: *
 15: _EXIT      equ          $ff00
 16: _SUPER     equ          $ff20
 17: *
 18: *etc.
 19: *
 20: iocs       equ          $0f
 21: G_VRAM     equ          $00c00000
 22:            .data
 23: SP_BUF:    dc.l         0
 24:            .text
 25:            clr.l        -(sp)
 26:            dc.w         _SUPER
 27:            addq.l       #4,sp
 28:            move.l       d0,SP_BUF
 29:            bsr          crtinit
 30:            bsr          crtwrite
 31:            bsr          keywait
 32:            bsr          pcmset
 33:            bsr          rec_dmaset
 34:            bsr          rec_pcmstart
 35:            bsr          wait_dma_end
 36:            bsr          pcmstop
 37:            bsr          keywait
 38:            bsr          pcmset
 39:            bsr          playdmaset
 40:            bsr          playpcmstart
 41:            bsr          wait_dma_end
 42:            bsr          pcmstop
 43:            move.l       SP_BUF,-(sp)
 44:            dc.w         _SUPER
 45:            addq.l       #4,sp
 46:            dc.w         _EXIT
 47: *
 48: * ｸﾞﾗﾌｨｯｸ画面を初期化(512×512、65535色)した後、
 49: * ｸﾞﾗﾌｨｯｸ表示をｲﾈｰﾌﾞﾙします
 50: *
 51: crtinit:
 52:            move.w       #12,d1
 53:            moveq.l      #_CRTMOD,d0
 54:            trap         #iocs
 55:            moveq.l      #_G_CLR_ON,d0
 56:            trap         #iocs
 57:            rts
 58: crtwrite:
 59:            movea.l      #G_VRAM,a0
 60:            moveq.l      #0,d0
 61:            move.w       #$ffff,d1
 62: crtwrite_loop:
 63:            move.b       d0,(a0)+
 64:            addq.l       #1,d0
 65:            dbra         d1,crtwrite_loop
 66:            rts
 67: *
 68: * ｷｰ入力があるまで待ちます
 69: *
 70: keywait:
 71:            moveq.l      #_B_KEYINP,d0
 72:            trap         #iocs
 73:            tst.l        d0
 74:            beq          keywait
 75:            rts
 76: *
 77: * DMAｾｯﾄｱｯﾌﾟ
 78: *
 79: CSR3            equ          $00e840c0
 80: CER3            equ          $00e840c1
 81: DCR3            equ          $00e840c4
 82: OCR3            equ          $00e840c5
 83: SCR3            equ          $00e840c6
 84: CCR3            equ          $00e840c7
 85: MTC3            equ          $00e840ca
 86: MAR3            equ          $00e840cc
 87: DAR3            equ          $00e840d4
 88: BTC3            equ          $00e840da
 89: BAR3            equ          $00e840dc
 90: NIV3            equ          $00e840e5
 91: EIV3            equ          $00e840e7
 92: MFC3            equ          $00e840e9
 93: CPR3            equ          $00e840ed
 94: DFC3            equ          $00e840f1
 95: BFC3            equ          $00e840f9
 96: FC_USER_DATA    equ          1
 97: FC_USER_PROG    equ          2
 98: FC_SUPER_DATA   equ          5
 99: FC_SUPER_PROG   equ          6
100: FC_INTA         equ          7
101: rec_dmaset:
102:    move.b        #$18,CCR3        * 動作ｽﾀｰﾄしない
103:    move.b        #$c0,DCR3        * あまり意味はない
104:    move.b        #$82,OCR3        * REQによる8ﾋﾞｯﾄ幅転送
105:    move.b        #FC_SUPER_DATA,MFC3
106:    move.b        #FC_SUPER_DATA,DFC3
107:    move.b        #$04,SCR3        * ﾒﾓﾘｰ側だけ増加
108:    move.l        #G_VRAM,MAR3     * ﾒﾓﾘｰｱﾄﾞﾚｽ
109:    move.l        #PCM_DATA,DAR3   * ﾃﾞｨｽﾃｨﾈｰｼｮﾝｱﾄﾞﾚｽ
110:    move.w        #$ffff,MTC3
111:    move.b        #$88,CCR3        * 動作ｽﾀｰﾄ
112:    rts
113: playdmaset:
114:    move.b        #$18,CCR3        * 動作ｽﾀｰﾄしない
115:    move.b        #$c0,DCR3        * あまり意味はない
116:    move.b        #$02,OCR3        * REQによる8ﾋﾞｯﾄ幅転送
117:    move.b        #FC_SUPER_DATA,MFC3
118:    move.b        #FC_SUPER_DATA,DFC3
119:    move.b        #$04,SCR3        * ﾒﾓﾘｰ側だけ増加
120:    move.l        #G_VRAM,MAR3     * ﾒﾓﾘｰｱﾄﾞﾚｽ
121:    move.l        #PCM_DATA,DAR3   * ﾃﾞｨｽﾃｨﾈｰｼｮﾝｱﾄﾞﾚｽ
122:    move.w        #$ffff,MTC3
123:    move.b        #$88,CCR3        * 動作ｽﾀｰﾄ
124:    rts
125: *
126: * ADPCM用PPIﾎﾟｰﾄｾｯﾄｱｯﾌﾟ
127: *
128: PCM_CMD     equ          $00e92001
129: PCM_DATA    equ          $00e92003
130: PCM_CTRL    equ          $00e9a005
131: PPI_CWR     equ          $00e9a007
132: PPI_CWD     equ          $92
133: pcmset:
134:            move.b       #PPI_CWD,PPI_CWR
135:            move.b       #$8,PCM_CTRL
136:            rts
137: *
138: * ADPCM動作開始
139: *
140: rec_pcmstart:
141:            move.b       #$4,PCM_CMD
142:            rts
143: playpcmstart:
144:            move.b       #$2,PCM_CMD
145:            rts
146: *
147: * ADPCM動作停止ｺﾏﾝﾄﾞ
148: *
149: pcmstop:
150:            move.b       #$1,PCM_CMD
151:            rts
152: *
153: * DMA転送が終了するのを待ちます
154: *
155: wait_dma_end:
156:            tst.w        MTC3
157:            bne          wait_dma_end
158:            rts
159:            .end
```

リスト3 CRTCテスト (OPEN.S)

```
  1: *-------------------------------------------
  2: *- ＣＲＴＣテスト（ひらけごま）
  3: *- パターンが表れた後、キーを叩くと・・・
  4: *-------------------------------------------
  5: *
  6: *IOCSｺｰﾙ
  7: *
  8: _G_CLR_ON  equ          $90
  9: _CRTMOD    equ          $10
 10: _B_KEYINP  equ          $00
 11: *
 12: *DOSｺｰﾙ
 13: *
 14: _SUPER     equ          $ff20
 15: _EXIT      equ          $ff00
 16: *
 17: *etc.
 18: *
 19: iocs       equ          $0f
 20: G_VRAM     equ          $00c00000
 21:            .data
 22: SPBUF:     dc.l         0
 23: GPIP       equ          $00e88001
 24: V_DISP     equ          $4
 25: HSADR      equ          $00e80004
 26: HEADR      equ          $00e80006
 27: VSADR      equ          $00e8000c
 28: VEADR      equ          $00e8000e
 29:            .text
 30:            clr.l        -(sp)
 31:            dc.w         _SUPER
 32:            addq.l       #4,sp
 33:            move.l       d0,SPBUF
 34:            bsr          crtinit
 35:            bsr          crtpatset
 36:            bsr          keywait
 37:            ori.w        #$0700,sr
 38:            move.w       #32,d0
 39:            move.w       #49,d1
 40:            move.w       #49,d2
 41:            move.w       #296,d3
 42:            move.w       #296,d4
 43: OPEN_SESAMI:
 44:            bsr          wait_v_disp
 45:            move.w       d1,HSADR
 46:            move.w       d2,HEADR
 47:            move.w       d3,VSADR
 48:            move.w       d4,VEADR
 49:            subq.l       #1,d1
 50:            addq.l       #1,d2
 51:            sub.w        #8,d3
 52:            add.w        #8,d4
 53:            move.w       #$ffff,d7
 54: OPEN_WAIT:
 55:            dbra         d7,OPEN_WAIT
 56:            dbra         d0,OPEN_SESAMI
 57:            andi.w       #$f8ff,sr
 58:            move.l       SPBUF,-(sp)
 59:            dc.w         _SUPER
 60:            addq.l       #4,sp
 61:            dc.w         _EXIT
 62: *
 63: * 垂直同期期間まで待ちます
 64: *
 65: wait_v_disp:
 66:            btst.b       #V_DISP,GPIP
 67:            bne          wait_v_disp
 68:            rts
 69: *
 70: * ｷｰ入力があるまで待ちます
 71: *
 72: keywait:
 73:            moveq.l      #_B_KEYINP,d0
 74:            trap         #iocs
 75:            tst.l        d0
 76:            beq          keywait
 77:            rts
 78: *
 79: * ｸﾞﾗﾌｨｯｸ画面を初期化(512×512、65535色)
 80: * した後、ｸﾞﾗﾌｨｯｸ表示をｲﾈｰﾌﾞﾙします
 81: *
 82: crtinit:
 83:            move.w       #12,d1
 84:            moveq.l      #_CRTMOD,d0
 85:            trap         #iocs
 86:            moveq.l      #_G_CLR_ON,d0
 87:            trap         #iocs
 88:            rts
 89: *
 90: * ｸﾞﾗﾌｨｯｸVRAMにﾊﾟﾀｰﾝを書き込みます
 91: *
 92: crtpatset:
 93:            movea.l      #G_VRAM,a0
 94:            moveq.l      #$0,d0
 95:            moveq.l      #3,d2
 96: patwrite_loop2:
 97:            move.w       #$ffff,d1
 98: patwrite_loop1:
 99:            move.w       d0,(a0)+
100:            addq.l       #$1,d0
101:            dbra         d1,patwrite_loop1
102:            dbra         d2,patwrite_loop2
103:            rts
104:            .end
```

リスト4 テキスト画面スクロールテスト (TXSCRL.S)

```
 1: *---------------------------------------------
 2: *- テキスト画面スクロールテスト              -
 3: *- キーを叩くとスクロール開始。              -
 4: *- 右からはみ出てきたところがすこし          -
 5: *- おかしくなっているのに注目!              -
 6: *---------------------------------------------
 7: *
 8: *IOCSｺｰﾙ
 9: *
10: _B_KEYINP  equ          $00
11: *
12: *DOSｺｰﾙ
13: *
14: _SUPER     equ          $ff20
15: _EXIT      equ          $ff00
16: *
17: *etc.
18: *
19: iocs       equ          $0f
20: G_VRAM     equ          $00c00000
21:            .data
22: SPBUF:     dc.l         0
23: GPIP       equ          $00e88001
24: V_DISP     equ          $4
25: TXT_X_SCRL equ          $00e80014
26: TXT_Y_SCRL equ          $00e80016
27:            .text
28:            clr.l        -(sp)
29:            dc.w         _SUPER
30:            addq.l       #4,sp
31:            move.l       d0,SPBUF
32:            bsr          keywait
33:            move.l       #0,d0           * X方向ｽｸﾛｰﾙ
34:            move.l       #0,d1           * Y方向ｽｸﾛｰﾙ
35:            move.l       #511,d2         * ｽｸﾛｰﾙ･ｶｳﾝﾀｰ
36: SCRL_LOOP1:
37:            ori.w        #$0700,sr       * 割り込み禁止
38:            bsr          wait_v_disp
39:            move.w       d0,TXT_X_SCRL
40:            move.w       d1,TXT_Y_SCRL
41:            andi.w       #$f8ff,sr       * 割り込み許可
42:            add.w        #1,d0
```

```
43:            add.w        #1,d1
44:            bsr          WAIT_A_MORMENT
45:            dbra         d2,SCRL_LOOP1
46:            move.l       #511,d2         * ｽｸﾛｰﾙ･ｶｳﾝﾀｰ
47: SCRL_LOOP2:
48:            ori.w        #$0700,sr
49:            bsr          wait_v_disp
50:            move.w       d0,TXT_X_SCRL
51:            move.w       d1,TXT_Y_SCRL
52:            andi.w       #$f8ff,sr
53:            sub.w        #1,d0
54:            sub.w        #1,d1
55:            bsr          WAIT_A_MORMENT
56:            dbra         d2,SCRL_LOOP2
57:            move.l       SPBUF,-(sp)
58:            dc.w         _SUPER
59:            addq.l       #4,sp
60:            dc.w         _EXIT
61: *
62: * 時間稼ぎ
63: *
64: WAIT_A_MORMENT:
65:            move.w       #$1fff,d7
66: WAIT_LOOP:
67:            dbra         d7,WAIT_LOOP
68:            rts
69: *
70: * 垂直同期期間まで待ちます
71: *
72: wait_v_disp:
73:            btst.b       #V_DISP,GPIP
74:            bne          wait_v_disp
75:            rts
76: *
77: * ｷｰ入力があるまで待ちます
78: *
79: keywait:
80:            moveq.l      #_B_KEYINP,d0
81:            trap         #iocs
82:            tst.l        d0
83:            beq          keywait
84:            rts
```

リスト5 プライオリティ制御テスト (PRI.S)

```
 1: *-------------------------------------------
 2: *- プライオリティ変更テスト                 -
 3: *-      キーを叩くとﾃｷｽﾄが表示され、        -
 4: *-      キーが叩かれる度プライオリティ      -
 5: *-      を入れ替えるようにしてみました      -
 6: *-------------------------------------------
 7: *
 8: *IOCSｺｰﾙ
 9: *
10: _G_CLR_ON  equ          $90
11: _CRTMOD    equ          $10
12: _B_KEYINP  equ          $00
13: *
14: *DOSｺｰﾙ
15: *
16: _PRINT     equ          $ff09
17: _SUPER     equ          $ff20
18: _EXIT      equ          $ff00
19: *
20: *etc.
21: *
22: iocs       equ          $0f
23: G_VRAM     equ          $00c00000
24: CR         equ          $d
25: LF         equ          $a
26:            .data
27: SPBUF:              dc.l        0
28: MSGBUF:    dc.b     CR,LF,CR,LF,CR,LF
29: dc.b '努　力　と　根　性　だ',CR,LF,CR,LF
30: dc.b 'ハ　ッ　ス　ル　、ハッスル',CR,LF,CR,LF
31: dc.b 'ギャフンと言わせてやる',CR,LF,CR,LF
32: dc.b 'コテンパンにたたきのめしてやる',CR,LF,CR,LF
33: dc.b '次回をお楽しみに。また来週も、見てね',CR,LF,CR,LF
34: dc.b '涙は 青春の 汗だ(とぎれとぎれに言う)',CR,LF,CR,LF
35: dc.b 'ひなたは体に毒ですよ(白い浴衣姿のお嬢様)',CR,LF,CR,LF
36: dc.b 'サ　ナ　ト　リ　ウ　ム',CR,LF
37: dc.w $0,0
38:
39: GPIP       equ          $00e88001
40: V_DISP     equ          $4
41: VCON_PRI   equ          $00e82500
42:            .text
43:            clr.l        -(sp)
44:            dc.w         _SUPER
45:            addq.l       #4,sp
46:            move.l       d0,SPBUF
47:            bsr          crtinit
48:            bsr          crtpatset
49:            bsr          keywait
50:            bsr          crtmesg
51:            bsr          keywait
52:            ori.w        #$0700,sr
53:            bsr          wait_v_disp
54:            move.w  #%00_01_10_00__00_01_10_11,VCON_PRI
55:            andi.w       #$f8ff,sr
56:            bsr          keywait
57:            ori.w        #$0700,sr
58:            bsr          wait_v_disp
59:            move.w  #%00_00_01_10__00_01_10_11,VCON_PRI
60:            andi.w       #$f8ff,sr
61:            move.l       SPBUF,-(sp)
62:            dc.w         _SUPER
```

▶やっと，X68000のローンが終わったぜ。長かった1年と3カ月，買いたいソフトも我慢してバイトしたかいがあった。さ，これからはソフトを思う存分買いまくるぞ!!　そして，68ユーザーの購買力を見せてやる。　藤山　健一 (20) 兵庫県

```
 63:           addq.l          #4,sp
 64:           dc.w            _EXIT
 65: *
 66: * 垂直同期期間まで待ちます
 67: *
 68: wait_v_disp:
 69:           btst.b          #V_DISP,GPIP
 70:           bne             wait_v_disp
 71:           rts
 72: *
 73: * ｷｰ入力があるまで待ちます
 74: *
 75: keywait:
 76:           moveq.l         #_B_KEYINP,d0
 77:           trap            #iocs
 78:           tst.l           d0
 79:           beq             keywait
 80:           rts
 81: *
 82: * ｸﾞﾗﾌｨｯｸ画面を初期化(512×512、256色)
 83: * ｸﾞﾗﾌｨｯｸ表示をｲﾈｰﾌﾞﾙします
 84: *
 85: crtinit:
 86:           move.w          #4,d1
 87:           moveq.l         #_CRTMOD,d0
 88:           trap            #iocs
 89:           moveq.l         #_G_CLR_ON,d0
 90:           trap            #iocs
 91:           rts
 92: *
 93: * ｸﾞﾗﾌｨｯｸVRAMにﾊﾟﾀｰﾝを書き込みます
 94: *
 95: crtpatset:
 96:           move.w          #50,d0                  * X座標
 97:           move.w          #50,d1                  * Y座標
 98:           move.w          #200,d2                 * Xｻｲｽﾞ
 99:           move.w          #200,d3                 * Yｻｲｽﾞ
100:           move.w          #3,d4                   * 色ｺｰﾄﾞ
101:           movea.l         #G_VRAM,a0              * VRAMのｱﾄﾞﾚｽ
102:           bsr             crt_write_box
103:           move.w          #100,d0                 * X座標
104:           move.w          #100,d1                 * Y座標
105:           move.w          #200,d2                 * Xｻｲｽﾞ
106:           move.w          #200,d3                 * Yｻｲｽﾞ
107:           move.w          #5,d4
108:           movea.l         #G_VRAM+$00080000,a0
109:           bsr             crt_write_box
110:           move.w          #150,d0                 * X座標
111:           move.w          #150,d1                 * Y座標
112:           move.w          #200,d2                 * Xｻｲｽﾞ
113:           move.w          #200,d3                 * Yｻｲｽﾞ
114:           move.w          #9,d4
115:           movea.l         #G_VRAM+$00100000,a0
116:           bsr             crt_write_box
117:           move.w          #200,d0                 * X座標
118:           move.w          #200,d1                 * Y座標
119:           move.w          #200,d2                 * Xｻｲｽﾞ
120:           move.w          #200,d3                 * Yｻｲｽﾞ
121:           move.w          #$7,d4
122:           movea.l         #G_VRAM+$00180000,a0
123:           bsr             crt_write_box
124:           rts
125: *
126: * 512×512ﾄﾞｯﾄ専用BOX
127: *
128: crt_write_box:
129:           ext.l           d0
130:           ext.l           d1
131:           lsl.l           #1,d0
132:           lsl.l           #8,d1       * 空いているﾚｼﾞｽﾀがない
133:           lsl.l           #2,d1       * ので2回に分けた
134:           add.l           d1,d0
135:           add.l           a0,d0
136:           sub.w           #1,d2
137:           sub.w           #1,d3
138:    crt_write_loopy:
139:           movea.l         d0,a0
140:           move.w          d2,d1
141:    crt_write_loopx:
142:           move.w          d4,(a0)+
143:           dbra            d1,crt_write_loopx
144:           add.l           #$400,d0
145:           dbra            d3,crt_write_loopy
146:           rts
147: *
148: * ﾃｷｽﾄを表示しておきます
149: *
150: crtmesg:
151:           pea             MSGBUF
152:           dc.w            _PRINT
153:           addq.l          #4,sp
154:           rts
155:           .end
```

リスト6 半透明の機能テスト（GTONE.S）

```
  1: *----------------------------
  2: *- 半透明機能テスト          -
  3: *-  キー入力1回目：          -
  4: *- プライオリティを変更      -
  5: *-  キー入力2回目：          -
  6: *- 半透明機能ON！            -
  7: *----------------------------
  8: *
  9: *IOCSｺｰﾙ
 10: *
 11: _G_CLR_ON  equu            $90
 12: _CRTMOD    equ             $10
 13: _B_KEYINP  equ             $00
 14: *
 15: *DOSｺｰﾙ
 16: *
 17: _SUPER     equ             $ff20
 18: _EXIT      equ             $ff00
 19: *
 20: *etc.
 21: *
 22: iocs       equ             $0f
 23: G_VRAM     equ             $00c00000
 24: CR         equ             $d
 25: LF         equ             $a
 26:            .data
 27: SPBUF:     dc.l            0
 28:
 29: GPIP       equ             $00e88001
 30: V_DISP     equ             $4
 31: VCON_PRI   equ             $00e82500
 32: VCON_SPEC  equ             $00e82600
 33:    .text
 34:    clr.l          -(sp)
 35:    dc.w           _SUPER
 36:    addq.l         #4,sp
 37:    move.l         d0,SPBUF
 38:    bsr            crtinit
 39:    bsr            crtpatset
 40:    bsr            keywait
 41:    ori.w          #$0700,sr
 42:    bsr            wait_v_disp
 43:    move.w         #%00_00_01_10__00_01_10_11,VCON_PRI
 44:    andi.w         #$f8ff,sr
 45:    bsr            keywait
 46:    ori.w          #$0700,sr
 47:    bsr            wait_v_disp
 48:    move.w         #$1eff,VCON_SPEC
 49:    andi.w         #$f8ff,sr
 50:    bsr            keywait
 51:    move.l         SPBUF,-(sp)
 52:    dc.w           _SUPER
 53:    addq.l         #4,sp
 54:    dc.w           _EXIT
 55: *
 56: * 垂直同期期間まで待ちます
 57: *
 58: wait_v_disp:
 59:           btst.b          #V_DISP,GPIP
 60:           bne             wait_v_disp
 61:           rts
 62: *
 63: * ｷｰ入力があるまで待ちます
 64: *
 65: keywait:
 66:           moveq.l         #_B_KEYINP,d0
 67:           trap            #iocs
 68:           tst.l           d0
 69:           beq             keywait
 70:           rts
 71: *
 72: * ｸﾞﾗﾌｨｯｸ画面を初期化(512×512、256色)
 73: * ｸﾞﾗﾌｨｯｸ表示をｲﾈｰﾌﾞﾙします
 74: *
 75: crtinit:
 76:           move.w          #4,d1
 77:           moveq.l         #_CRTMOD,d0
 78:           trap            #iocs
 79:           moveq.l         #_G_CLR_ON,d0
 80:           trap            #iocs
 81:           rts
 82: *
 83: * ｸﾞﾗﾌｨｯｸVRAMにﾊﾟﾀｰﾝを書き込みます
 84: *
 85: crtpatset:
 86:           move.w          #50,d0                  * X座標
 87:           move.w          #50,d1                  * Y座標
 88:           move.w          #200,d2                 * Xｻｲｽﾞ
 89:           move.w          #200,d3                 * Yｻｲｽﾞ
 90:           move.w          #7,d4                   * 色ｺｰﾄﾞ
 91:           movea.l         #G_VRAM,a0              * VRAMｱﾄﾞﾚｽ
 92:           bsr             crt_write_box
 93:           move.w          #100,d0                 * X座標
 94:           move.w          #100,d1                 * Y座標
 95:           move.w          #200,d2                 * Xｻｲｽﾞ
 96:           move.w          #200,d3                 * Yｻｲｽﾞ
 97:           move.w          #5,d4
 98:           movea.l         #G_VRAM+$00080000,a0
 99:           bsr             crt_write_box
100:           move.w          #150,d0                 * X座標
101:           move.w          #150,d1                 * Y座標
102:           move.w          #200,d2                 * Xｻｲｽﾞ
103:           move.w          #200,d3                 * Yｻｲｽﾞ
104:           move.w          #3,d4
105:           movea.l         #G_VRAM+$00100000,a0
106:           bsr             crt_write_box
107:           move.w          #130,d0                 * X座標
108:           move.w          #30,d1                  * Y座標
109:           move.w          #100,d2                 * Xｻｲｽﾞ
110:           move.w          #350,d3                 * Yｻｲｽﾞ
111:           move.w          #$b,d4
112:           movea.l         #G_VRAM+$00180000,a0
113:           bsr             crt_write_box
114:           rts
115: *
116: * 512×512ﾄﾞｯﾄ専用BOX
117: *
118: crt_write_box:
119:           ext.l           d0
120:           ext.l           d1
121:           lsl.l           #1,d0
122:           lsl.l           #8,d1   * 空いているﾚｼﾞｽﾀが
123:           lsl.l           #2,d1   * ないので2回に分けた
124:           add.l           d1,d0
125:           add.l           a0,d0
126:           sub.w           #1,d2
127:           sub.w           #1,d3
128: crt_write_loopy:
129:           movea.l         d0,a0
130:           move.w          d2,d1
131: crt_write_loopx:
132:           move.w          d4,(a0)+
133:           dbra            d1,crt_write_loopx
134:           add.l           #$400,d0
135:           dbra            d3,crt_write_loopy
136:           rts
137:           .end
```

▶ふと気づくと，私はBASIC，SLANG，64180アセンブラ，68000アセンブラしか使えない人間になっていた。Cをやろうと，C CompilerPRO-68Kを買ったのだが「BC.X」のおかげでアセンブラかBASICしか使わなくなった。こんな私はC言語一色の世の中で生きていけるでしょうか。
田中　智樹（17）京都府

リスト7　スプライトテスト（SPDEF.S）

```
  1: *----------------------------
  2: *- スプライトテスト          -
  3: *-  1回目のキー入力：        -
  4: *- スプライトの定義と表示    -
  5: *-  2回目のキー入力：        -
  6: *- スプライトの移動          -
  7: *----------------------------
  8: *
  9: *IOCSｺｰﾙ
 10: *
 11: _G_CLR_ON  equ           $90
 12: _CRTMOD    equ           $10
 13: _B_KEYINP  equ           $00
 14: *
 15: *DOSｺｰﾙ
 16: *
 17: _SUPER     equ           $ff20
 18: _EXIT      equ           $ff00
 19: *
 20: *etc.
 21: *
 22: iocs       equ           $0f
 23: G_VRAM     equ           $00c00000
 24: CR         equ           $d
 25: LF         equ           $a
 26:            .data
 27: SPBUF:     dc.l          0
 28: PALDAT     dc.w          0
 29:            dc.w          21140
 30:            dc.w          32
 31:            dc.w          62
 32:            dc.w          1024
 33:            dc.w          1984
 34:            dc.w          1056
 35:            dc.w          2046
 36:            dc.w          32768
 37:            dc.w          63488
 38:            dc.w          32800
 39:            dc.w          63550
 40:            dc.w          33792
 41:            dc.w          65472
 42:            dc.w          44394
 43:            dc.w          65534
 44: PATDAT     dc.b    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
 45:            dc.b    0,0,0,0,0,11,11,11,0,11,11,11,0,0,0,0
 46:            dc.b    0,0,0,0,0,11,11,11,0,11,11,11,0,0,0,0
 47:            dc.b    0,0,0,0,0,11,11,11,0,5,11,5,0,0,0,0
 48:            dc.b    0,0,0,0,0,11,11,11,0,11,11,11,0,0,0,0
 49:            dc.b    0,0,0,0,0,11,11,11,11,11,11,11,0,0,0,0
 50:            dc.b    0,0,0,0,0,11,11,11,0,11,11,11,0,0,0,0
 51:            dc.b    0,0,0,0,0,11,10,11,0,11,11,11,0,0,0,0
 52:            dc.b    0,0,0,0,0,11,10,11,0,11,11,11,0,0,0,0
 53:            dc.b    0,0,0,0,0,11,10,11,0,11,11,11,0,0,0,0
 54:            dc.b    0,0,0,0,0,11,10,11,0,11,11,11,0,0,0,0
 55:            dc.b    0,0,0,0,0,11,10,11,11,11,11,11,0,0,0,0
 56:            dc.b    0,0,0,0,0,11,10,11,0,11,11,11,0,0,0,0
 57:            dc.b    0,0,0,0,0,11,11,11,0,11,11,11,0,0,0,0
 58:            dc.b    0,0,0,0,0,13,13,13,7,13,13,13,0,0,0,0
 59:            dc.b    0,0,0,0,0,13,14,13,7,13,14,13,0,0,0,0
 60: GPIP       equ     $00e88001
 61: V_DISP     equ     $4
 62: VCON_PRI   equ     $00e82500
 63: VCON_SPEC  equ     $00e82600
 64: SP_SCRL    equ     $00eb0000
 65: BG_CTRL    equ     $00eb0808
 66: PAT_TABLE  equ     $00eb8000         * ﾊﾟﾀｰﾝ
 67: PAL_TABLE  equ     $00e82200         * ﾊﾟﾚｯﾄ
 68:            .text
 69:            clr.l   -(sp)
 70:            dc.w    _SUPER
 71:            addq.l  #4,sp
 72:            move.l  d0,SPBUF
 73:            bsr     crtinit
 74: *          bsr     crtpatset
 75:            bsr     keywait
 76:            bsr     pal_init
 77:            bsr     sp_init
 78:            bsr     sp_def
 79:            bsr     sp_disp
 80:            movea.l #SP_SCRL,a0
 81:            move.w  #100,(a0)         * X座標
 82:            move.w  #100,2(a0)        * Y座標
 83:            move.w  #$0100,4(a0)      * 反転,色とｽﾌﾟﾗｲﾄのｺｰﾄﾞ
 84:            move.w  #$1,6(a0)         * ﾌﾟﾗｲｵﾘﾃｨ& SP_ON(1)
 85:            bsr     keywait
 86:            move.w  #255,d0
 87:            moveq.l #0,d1
 88: sp_move_loop:
 89:            move.w  d1,(a0)           * X座標
 90:            move.w  d0,2(a0)
 91:            addq.l  #1,d1
 92:            bsr     wait_delay
 93:            dbra    d0,sp_move_loop
 94:            move.l  SPBUF,-(sp)
 95:            dc.w    _SUPER
 96:            addq.l  #4,sp
 97:            dc.w    _EXIT
 98: *
 99: * ちょっとした時間稼ぎ
100: *
101: wait_delay:
102:            move.w            #$1fff,d2
103: wait_delay_loop:
104:            dbra              d2,wait_delay_loop
105:            rts
106: *
107: * ｽﾌﾟﾗｲﾄ表示ON
108: *
109: sp_disp
110:            ori.w   #$0700,sr
111:            bsr     wait_v_disp
112:            move.w  #$0200,BG_CTRL
113:            move.w  #%00_01_00_10__00_01_10_11,VCON_PRI
114:            move.w  #$007f,VCON_SPEC  * SP_DISP(1)相当
115:            andi.w  #$f8ff,sr
116:            rts
117: *
118: * ｽﾌﾟﾗｲﾄの座標などの初期化
119: *
120: sp_init:
121:            movea.l #SP_SCRL,a0
122:            move.w  #127,d0
123: sp_off_loop:
124:            move.w  #0,(a0)+          * X座標
125:            move.w  #0,(a0)+          * Y座標
126:            move.w  #$0100,(a0)+      *反転,色とｽﾌﾟﾗｲﾄのｺｰﾄﾞ
127:            move.w  #$0,(a0)+         * ﾌﾟﾗｲｵﾘﾃｨ& SP_OFF(n)
128:            dbra    d0,sp_off_loop
129:            rts
130: *
131: * ﾊﾟﾚｯﾄの初期化(0番はﾃｷｽﾄ用なのでいじらない)
132: *
133: pal_init:
134:            movea.l         #PAL_TABLE+$20,a0
135:            move.w          #14,d1
136: palette_setup2:
137:            lea.l           PALDAT,a1
138:            move.w          #15,d0
139: palette_setup1:
140:            move.w          (a1)+,(a0)+
141:            dbra            d0,palette_setup1
142:            dbra            d1,palette_setup2
143:            rts
144: *
145: * ｽﾌﾟﾗｲﾄﾊﾟﾀｰﾝ定義
146: *
147: sp_def:
148:            movea.l         #PAT_TABLE,a2
149:            lea.l           PATDAT,a1
150:            move.w          #15,d0
151: pattern_setup2:
152:            move.w          #1,d2
153:            move.l          a2,a0
154: pattern_setup1:
155:            moveq.l         #0,d1
156:            move.b          (a1)+,d1
157:            lsl.l           #4,d1
158:            or.b            (a1)+,d1
159:            lsl.l           #4,d1
160:            or.b            (a1)+,d1
161:            lsl.l           #4,d1
162:            or.b            (a1)+,d1
163:            lsl.l           #4,d1
164:            or.b            (a1)+,d1
165:            lsl.l           #4,d1
166:            or.b            (a1)+,d1
167:            lsl.l           #4,d1
168:            or.b            (a1)+,d1
169:            lsl.l           #4,d1
170:            or.b            (a1)+,d1
171:            move.l          d1,(a0)
172:            adda.l          #$40,a0
173:            dbra            d2,pattern_setup1
174:            adda.l          #4,a2
175:            dbra            d0,pattern_setup2
176:            rts
177: *
178: * 垂直同期期間まで待ちます
179: *
180: wait_v_disp:
181:            btst.b          #V_DISP,GPIP
182:            bne             wait_v_disp
183:            rts
184: *
185: * ｷｰ入力があるまで待ちます
186: *
187: keywait:
188:            moveq.l         #_B_KEYINP,d0
189:            trap            #iocs
190:            tst.l           d0
191:            beq             keywait
192:            rts
193: *
194: * ｸﾞﾗﾌｨｯｸ画面を初期化(256×256)し、
195: * ｸﾞﾗﾌｨｯｸ表示をｲﾈｰﾌﾞﾙします
196: *
197: crtinit:
198:            move.w          #2,d1
199:            moveq.l         #_CRTMOD,d0
200:            trap            #iocs
201:            moveq.l         #_G_CLR_ON,d0
202:            trap            #iocs
203:            rts
204: *
205: * ｸﾞﾗﾌｨｯｸVRAMにﾊﾟﾀｰﾝを書き込みます
206: *
207: crtpatset:
208:            move.w          #25,d0            * X座標
209:            move.w          #25,d1            * Y座標
210:            move.w          #100,d2           * Xｻｲｽﾞ
211:            move.w          #100,d3           * Yｻｲｽﾞ
212:            move.w          #7,d4             * 色ｺｰﾄﾞ
213:            movea.l         #G_VRAM,a0        * VRAMのｱﾄﾞﾚｽ
214:            bsr             crt_write_box
215:            move.w          #50,d0            * X座標
216:            move.w          #50,d1            * Y座標
217:            move.w          #100,d2           * Xｻｲｽﾞ
218:            move.w          #100,d3           * Yｻｲｽﾞ
219:            move.w          #5,d4
220:            movea.l         #G_VRAM+$00080000,a0
221:            bsr             crt_write_box
222:            move.w          #75,d0            * X座標
223:            move.w          #75,d1            * Y座標
224:            move.w          #100,d2           * Xｻｲｽﾞ
225:            move.w          #100,d3           * Yｻｲｽﾞ
226:            move.w          #3,d4
227:            movea.l         #G_VRAM+$00100000,a0
228:            bsr             crt_write_box
229:            move.w          #100,d0           * X座標
230:            move.w          #100,d1           * Y座標
231:            move.w          #100,d2           * Xｻｲｽﾞ
232:            move.w          #100,d3           * Yｻｲｽﾞ
233:            move.w          #$b,d4
234:            movea.l         #G_VRAM+$00180000,a0
235:            bsr             crt_write_box
236:            rts
237: *
238: * 256×256ﾄﾞｯﾄ専用BOX
239: *
240: crt_write_box:
241:            ext.l           d0
242:            ext.l           d1
243:            lsl.l           #1,d0
244:            lsl.l           #8,d1   * 空いているﾚｼﾞｽﾀがない
245:            lsl.l           #1,d1   * ので2回に分けた
246:            add.l           d1,d0
247:            add.l           a0,d0
248:            sub.w           #1,d2
249:            sub.w           #1,d3
250: crt_write_loopy:
251:            movea.l         d0,a0
252:            move.w          d2,d1
253: crt_write_loopx:
254:            move.w          d4,(a0)+
255:            dbra            d1,crt_write_loopx
256:            add.l           #$400,d0
257:            dbra            d3,crt_write_loopy
258:            rts
259:            .end
```

Oh!X readers'ぎゃらりい

久々に読者の皆さんから届いたカラーイラストを一挙掲載します。夏物の絵が多いけど，季節はずれだなんていじめないでネ。

▲小林　貴洋　（千葉県）

▲伊藤　浩克　（香川県）

▲安川　実　（愛知県）

▲伊藤　大地　（東京都）

▲大野　真実　（静岡県）
あまりの細かさ凄まじさに圧倒されてしまい，大判サイズ（縮小率60%）で載せちゃった。

▲白沢　桂一　（千葉県）

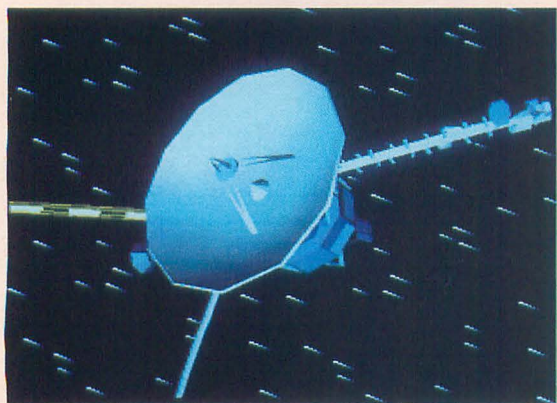
▲島康　太郎　（千葉県）

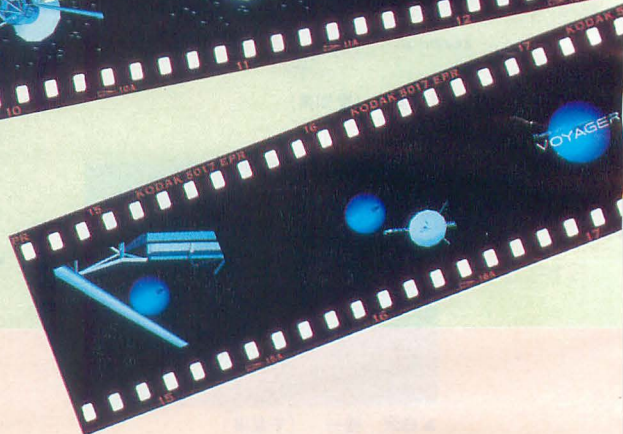
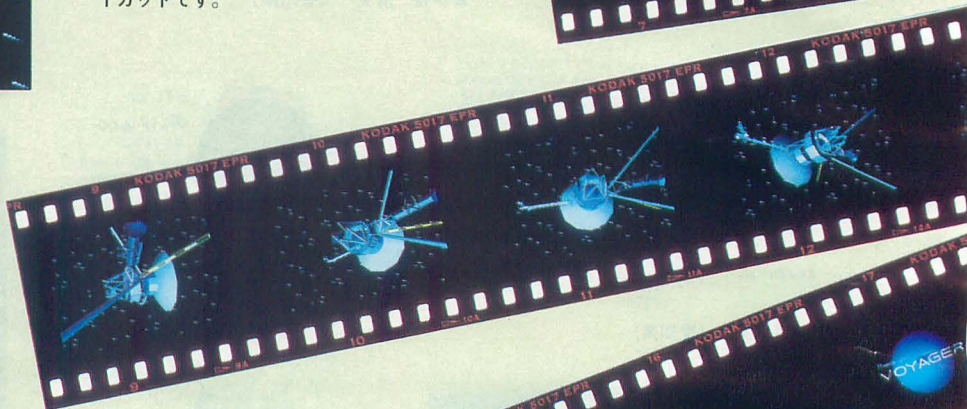
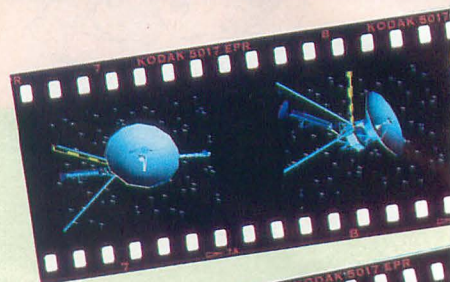
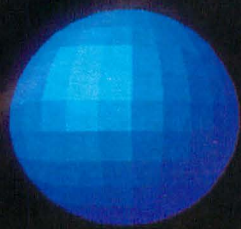
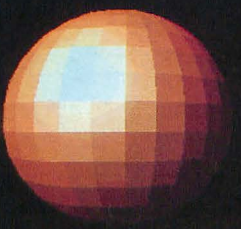
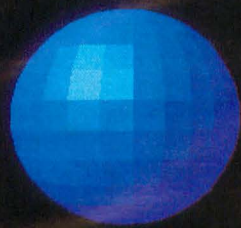
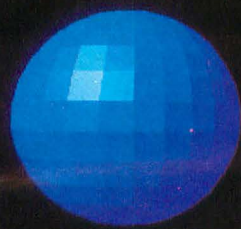
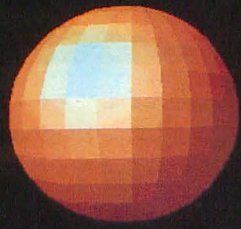
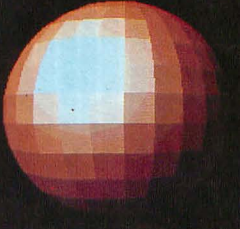
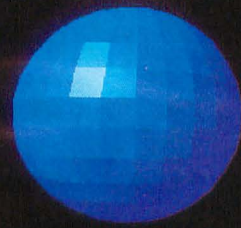
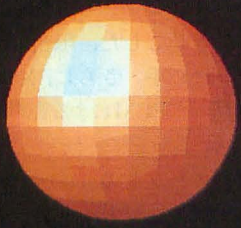
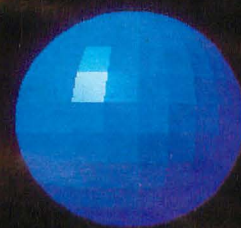
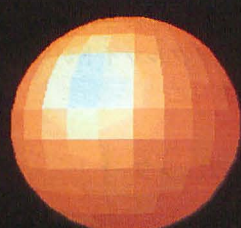
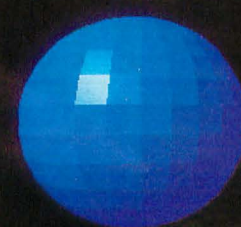
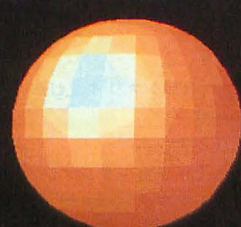
▲山崎　潤一
以前本誌でも紹介したパソコンクラブWATER発行の「シェイク」。なんとVol. 7の表紙を飾るのは山崎君のカラーCGなのです。というわけで山崎君，Oh！Xにもイラストお願いね。

▲丸藤　俊之　（神奈川県）

▲高橋　弘幸　（神奈川県）

次回の「readers'ぎゃらりぃ」は来年の3月号（締め切りは1月20日ごろ）を予定しています。CGやイラストの年賀状をお待ちしています。

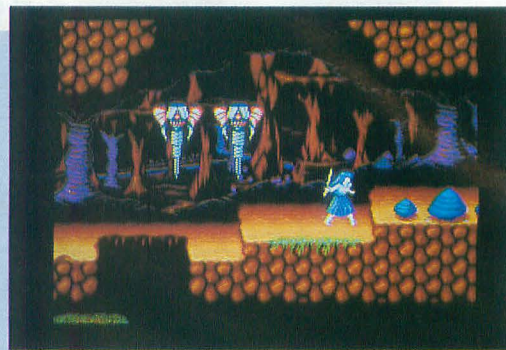
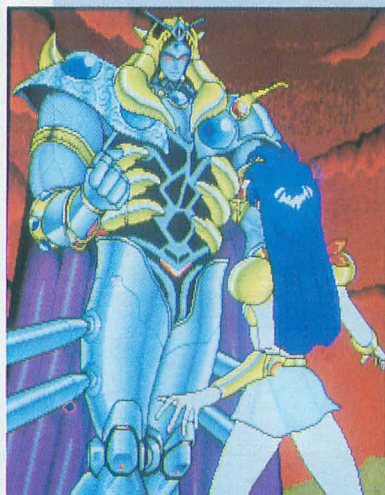
Oh!X Graphic Gallery

DōGA・CGアニメーション講座

まずは，プロジェクトチームDōGAの最新作「Thank you VOYAGER」をご紹介しましょう。先月号で紹介したスムーズシェーディングをボイジャーの一部に，そしてマッピングを海王星に使っています。続いて，「ただいま会員募集中」の鳥取大学の制作による，今年度「アマチュアCGアニメーションコンテスト」用作品「ART-3(仮名)」の1カットです。

MZ-2500グラフィックエディタ作成講座

画餅システムもいよいよ完成！　最終回の今月はしめくくりにふさわしいハードコピールーチンです。大判小判の2種類を用意いたしました。下は画餅の全ウィンドウを開いたところ。さあ，これでもうX68000なんかこわくない？

永久保存版「アトリビュート早見表」

＊これで君のマニュアルにもカラーページができる！ というわけで、このページを切り取って、DōGA CGAシステムのマニュアルの表紙の裏に張り付けてご利用ください。うらのページの方ゴメンナサイ。

SOFTWARE information

夢幻戦士ヴァリスⅡ
写真はいずれも，正真正銘X68000で撮影したオリジナル画面。鮮やかにスクロールするゲーム画面に加えて，全編を彩る豊富なビジュアルシーンが魅力。

話題のソフトウェア

この年末に備えて発売予定ソフトが続々と名乗りを上げてきました。というわけで今月はスペースがないので，さっそく，本題に突入しましょう。

まずは，日本テレネットの**夢幻戦士ヴァリスⅡ**。セーラー服の優子ちゃんが黒目を40パーセントもパワーアップして帰ってきました。新たに描き起こした，緻密で大胆なグラフィックが決め手。さらに，X68000版では全編にわたってPCMによる音声合成も聴けるという話。

続いて，アルシスソフトの3Dのシューティングアクションは名称が**ナイトアームズ**になりました。また，縦横スクロールの迷路での戦闘シーンの画面をご覧にいれましょう。シューティングではこのアームズとシステムサコムの**メタルサイト**が期待大といったところですね。

一方，前号でもお伝えした**フラッピー2**ですが，今回はさまざまな仕掛けも加わって，かなりアクション色の強い構成になっています。火山の面で“火ぐま”が登場するところなどを見ると，パワフルまあじゃんで見せたデービー独特のノリも持ち込んでくれている様子。期待して待とうね。

さてさて，エアホッケーといえばピンとくる人もいるかもしれませんが，ブローダーバンドの**シャッフル・パック・カフェ**はちょっとばかり期待してもらわなくちゃなりません。スピーディかつ壮快なゲームで，シンプルだがゲーム性が抜群なのです。

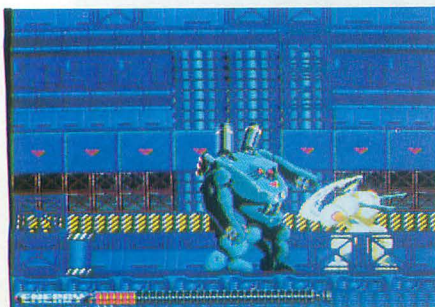
そのほか，ズームの新作RPGは**ラグーン**だ！　とか，工画堂スタジオがいよいよ**シュヴァルツシルト**でX68000に参入！　とか，ゲーム以外では，サン・ミュージカル・サービスが新しいグラフィックツールを作っているぞ！　とか，ツァイトがNEWS用のDTPシステムDear Layouterを X68000 に移植するぞ！　とか，シャープの**サイバーノート**ってなに？　とかは来月をお楽しみに。

RPGの巻き返し，スタート！

		(前回順位)
1	ジェノサイド	1
2	ファンタジーゾーン	3
3	アフターバーナー	2
4	テトリス	4
5	サバッシュ	5
6	リングマスター	―
7	ソーサリアン(追加シナリオ含む)	10
8	スタークルーザー	9
9	ローグ・アライアンス	―
10	アドヴァンスト・ファンタジアン	7

あららら。今月もアフターバーナーとジェノサイドの熾烈なトップ争いを期待して集計したら，アフターバーナーがトップどころかなんと3位転落。どうやらみんなひと通り遊んだところにゲーム特集でいろんなソフトが紹介されたので，票がそっちに散ってしまったようです。

サバッシュに続いてリングマスターとローグ・アライアンスの2本のRPGがランクインを果たしました。2本とも本格派RPGファンの心をしっかりつかんでいるようで，特にリングマスターにはテーブルトーク的なシステムを推薦理由に挙げる人が多いですね。まだまだ出たばっかりだし，ローグ・アライアンスもX1版が発売されたので，得票の伸びが楽しみ。

さて全体を見ると，実はテトリス以外はみ～んなRPG的要素が強いものだ。信長の野望や三国志といったシミュレーション勢ももうひとつ伸びに欠けているのが実情です。　(浦)

新作ソフト情報

☆……10月1日現在発売中　★……近日発売予定

☆ローグ・アライアンス

先月のゲーム特集にも紹介されたローグ・アライアンスの移植版が発売となった。原作はSSI社の「リアルム・オブ・ダークネス」というゲーム。SSIというのは斬新なゲームデザインで知られており，このゲームにも地上部分にコマンド入力が使われているなど，その手腕が存分に発揮されている。X68000と同じくマウス対応で，アイコンからの入力も可能だ。本格的ロールプレイとして，長く楽しめる。

X1/X1turbo用　5″2D版4枚組 9,800円
スタークラフト　☎03(988)2988

☆ウルティマⅡ

大魔王モンディンはついに倒され，ブリタニアに平和が戻ったかに見えた。しかし，悪の根は絶えてはいなかったのである。今回プレイヤーの行手にたちはだかるのは，大魔王モンディンの愛弟子，魔女ミナクス。太古の伝説の時代から未来の世界までと，時間までも股にかけたクエストが繰り広げられる。さらに宇宙船を手に入れることによって8つの惑星を旅することまでできるのだ。ウイザードリィとともにRPGの巨頭と並び称される名作の第2作。

X1/X1turbo　5″2D版3枚組 7,800円
ポニーキャニオン　☎03(221)3161

★アルフェイム

主人公，池内良の通うサンルート高校は，偏差値の高い順にA，B，Cの各クラスに分けられている。落ちこぼれのCクラスで楽しく過ごしていた良だったが，親友であった奥内武が突然自殺してしまう。この自殺に納得のいかなかった良はこの事件の捜査を開始したが，その後も校内で不可思議な事件が続くなか，捜査途中で謎の転校生の存在が浮かび上がってきた。

アニメーションとフルウィンドウを駆使してストーリーは語られていく。学園を舞台としたSFミステリーだ。

X68000用　5″2HD版4枚組 9,800円
ザインソフト　☎0794(31)7453

★やじうまペナントレース

野球ゲームのなかでもひときわ異彩を放つのがこの「やじうまペナントレース」。2人でゲームのみを楽しむほかに，選手として自分の役割をこなしながら同時に監督として采配を振るうこともできるペナントレースモードがある。プレイヤーは希望の球団に投手か野手として入団。試合の前には練習をして走力，体力といったパラメータを上げ，試合では自分の出番以外では選手交替・サインプレーなどを指示。見事リーグ優勝すれば日本シリーズに出場。さらに日本一になると大リーグとの対戦が待っている。X68000版では1989年度のプロ野球最新データを使用。

X68000用　5″2HD版2枚組 7,800円
ビクター音楽産業　☎03(423)7901

★斬(ZAN)——陽炎の時代——

X68000に，戦国シミュレーションが登場。内政，外交，諜報，そして行軍に手腕を振るい，全国の統一を目指すのだ。特長は，コマンド数をあえて減らし，斬新なシステムによって「見せる」ゲームに仕上げた点だ。自然なマップを描けるノーライン・ヘクス，戦闘時の天候や相手の数によって相手が見え隠れするナチュラル・ブラインド・サーチ，ゲーム進行を歴史年表にまとめてくれる歴史の書システムなどが導入されている。ウルフ独特のデザインを施した“斬”，いよいよ登場！

X68000用　5″2HD版 7,800円
ウルフチーム　☎03(5273)4795

★夢幻戦士ヴァリスⅡ

夢幻戦士ヴァリスとして，夢幻王ログレスを倒した優子。しかしログレス亡きあとのヴェガンティィは残忍王メガスの指揮のもと統一され，再びリアリティ（現実世界）への侵攻を再開した。ヴァリス続編の登場だ。パワーアップ型のシューティングゲームというスタイルは前作のままだが，今回は武器が5種類4レベル，防具が8種類となり，状況に応じて持ち換えが可能になった。防具の付け換え画面では着せ替え的な楽しみ方も？　内容も強制スクロールやトラップと，よりバリエーションに富んだ展開を見せてくれる。

X68000用　5″2HD版4枚組 9,800円
日本テレネット　☎03(268)1159

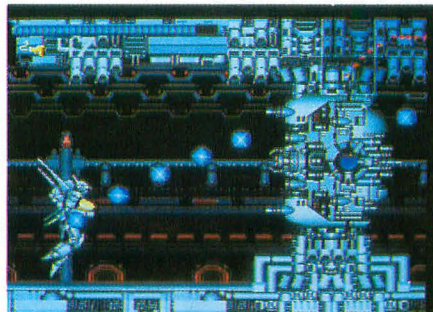
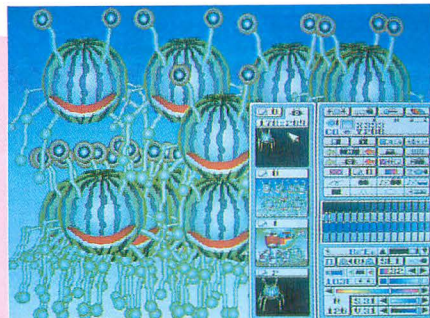
★TELENET MUSIC BOX

アメリカン・トラックからヴァリスⅡまで，テレネット作品の音楽全174曲入りという豪華なBGM集。X68000に出ていないソフトのBGMもOPM用にアレンジされて収録されているからファンにはこたえられないだろう。機能的にも，シャッフルプレイやリピートプレイなどのCDプレイヤー的なものから，曲順を選んでセーブするなどのコンピュータならではのものまで用意されている。全部聴くだけでも大変なボリュームだ。

X68000用　5″2HD版 3,980円
日本テレネット　☎03(268)1159

★フラッピー2・ブルースターの復活

エビーラやユニコーンをかわしてブルーストーンをブルーエリアに運ぶアクションタイプのパズルゲーム。あのフラッピー独特の画面（50面）だけではなく，5種類の“わーるど”が設けられ，それぞれ固有の仕掛けや敵キャラが用意される。大きくなって表情豊かになったキャラクターの動きが特長で，坂をすべり下りたり，壁に張りついて横歩きをしたりと，見ているだけでも楽しい。

X68000用　5″2HD版 予価 8,800円
デービーソフト　☎011(807)6700

サン・ミュージカル・サービスが開発中のグラフィックツール

★エメラルドドラゴン

ドラゴンのアトルシャンとともに育てられたタムリンが，魔軍の攻略を受けて壊滅状態だった故郷イシュ・バーンを救う，アクションタイプのRPG。タムリン，アトルシャンのほかにイベントごとにゲストキャラを加え，5人のパーティを組んで冒険に出る。戦闘はファンタジアンタイプで，アトルシャン以外はコンピュータにも任せられる。豊富なビジュアルで語られる重層なシナリオ。魔軍の目的と正体は？　冒険の始まる日は近い。

X68000用　5″2HD版 予価 9,800円
バショウハウス　☎03(219)6123

★シャッフル・パック・カフェ

Macintosh用に発売されていたエアホッケーゲーム「シャッフル・パック・カフェ」がX68000に移植された。とてつもない速さでパックがすっ飛ぶ気の抜けないゲームだ。舞台は宇宙の酒場で，対戦者は強さも個性もみんな違う銀河の流れ者が11人。好みの相手と対戦するモードもあり，自分の腕に合わせて，パドルの大きさや反発力などを細かく調整できる。

X68000用　5″2HD版2枚組 7,800円
ブロダーバンドジャパン　☎03(341)1131

ナイトアームズ

メタルサイト

フラッピー2

シャッフル・パック・カフェ

GAME REVIEW

今月はX1用にはRPG「ROGUE ALLIANCE」を,そしてX68000用にはテーブルゲーム「天九牌」とタケルソフトのシューティングゲーム「C-ON-Z」をお届けします。海外の移植もの,中国古来のゲーム,雑誌の投稿作品と,それぞれ個性が強いですね。

ROGUE ALLIANCE

海外ものの移植だが,まさにRPGの見本といえる作品。シナリオもよくできており,アドベンチャーモードも楽しめる。

▶X68000版に続いて,X1版まで発売されてしまったローグアライアンスです。海外の移植もの RPG として好評でした。それがなんと! シナリオからクエストの迷路にいたるまで,X68000版とそっくり同じように移植されています。そう,違っているとすれば,解像度によるものと音楽ぐらいでしょうか。なにしろ,X1のくせに(失礼)マウスまでサポートされているんですから。X1ユーザーには,ぜひプレイしてもらいたいソフトです。

紹介しておきますと,このローグアライアンスはいわゆる本格派 RPG でして,けっしてハデさはありませんが,奥の深さを感じさせるシナリオです。移植ものにありがちな,グラフィックよりもテキスト(文字)重視ソフトですが,意外に新鮮です。随所で,日本とは違う文化に接近遭遇するでしょう。詳しくは,10月号のX68000版の解説を読んでみて!! ほとんどそのまま通用しますよ。

熱中度▶▶▶▶▶▷▷ (亀)

▶ローグアライアンスの世界は広く,険しく,いったいいくつのクエストが用意されているのか,それは冒険を終了した者のみが知ることができる。10月号でX68000版が紹介されていたのでゲームの内容については触れないけど,嬉しいことに無印X1でもこのゲームはプレイすることができるんです。これだけ強者のRPGならなかなかゲームが発売されなくても,その間もたっぷり楽しませてくれることになるんじゃないかな。うーん,今年のX1ユーザーには一番遊べるソフトかもしれない。えっなになに,マウスでアイコンを選択することができるんだって。あらほんとだ,これは便利だけど,マウスがないとキーボードから文字を入力して,アイコンを選択しなくちゃならないのね。それもアイコンに入力する文字がマニュアルにしか書いてないから,最初はどの文字を入力するのかわからないところが片手落ちだな。それ以外はまったく不満なし! 合格点をあげちゃいます。

熱中度▶▶▶▶▶▷▷ (H.K.)

X1/X1turbo用 5″2D版3枚組 9,800円(税別)
(2ドライブ専用)
スタークラフト ☎03(988)2988

天九牌

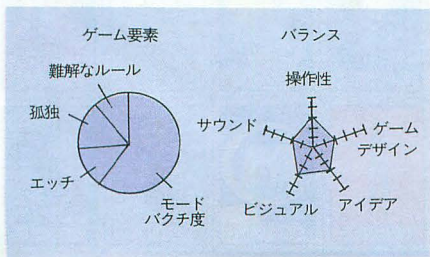
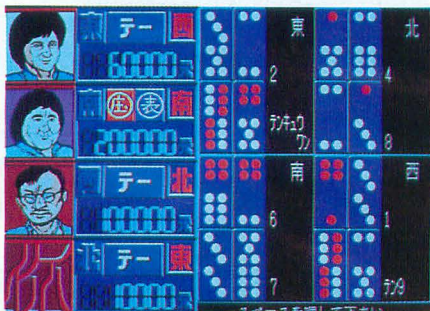
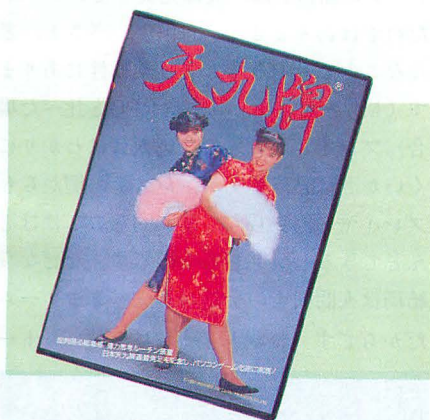
中国6000年の歴史が生んだテーブルゲームのパソコンゲーム版。ルールはちょっと覚えづらいが,なかなか面白いゲームだ。

▶ドミノを使って4人で遊ぶテーブルゲームである。もちろん頭数が足りない分は,コンピュータが肩代わりする。ドミノ倒しばかりが有名なドミノだが,本来はトランプと同じく娯楽用具の一種なのだ。「天九牌」とは,数あるドミノの遊び方のうちのひとつで,ポーカーに似ている。簡単にいうと,賭け金を決めひとり当たり4つの牌を配り,その役を計算して勝敗を競うというゲームだ。ポーカーよりも,さらに運まかせの要素が強く,ほとんど丁か半かの世界である。よって,テクニックを蓄積をする隙がなく,ひとりで遊ぶには虚しい。が,そこんとこは「桃源郷編――可愛い女の子たちと遊びましょ」を選択することによってごまかせるだろう。余計虚しい気分になるかもしれないが……。あと1.5倍ほどスピードアップし,軽快なBGMでもつけてくれれば,も

っとテンポよく遊べただろうに，と思うのだが。

熱中度　▶▶▶▶▶▷▷　(お)

▶ちょっとやると，自分のすることが少ない淡白なゲームなんですが，本気でコンピュータを打ち負かそうと思ったらなかなかハマります，このゲーム。基本的にはブラックジャックのように高い役の出そうなカードの山に賭けて親と張り合うんですが「あの人はここに賭けたけど，僕は親のほうが勝つと思う」とか「親に財政援助しよう」という賭け方があって，状況に合わせたシビアな選択を要求されます。

プログラムも優秀ではないけどまぁ及第点だし，脱いでくれる女の子も（あれ，なにこだわってんだ，あははは）アクのないアニメ顔で好感が持てます。と，聞くと面白そうでしょ。面白いんだけど，そこにいたるまでの時間が長いこと長いこと。天九牌の説明にもっとまとまったページを取ってほしかった。一時はルールを永遠に理解できないのかと思ってその辺をさすらってしまいました。せめてプレイヤーの努力に頼らないマニュアルを書いてほしいもんです。

熱中度▶▶▶▶▷▷▷　(H.U.)

X68000用　5″2HD版2枚組　6,800円（税別）
スタジオパンサー　☎03(798)2760

C-ON-Z

ステージ数は全部で5つ。パワーアップやスピードアップを取りつつ進んでいくのだがちょっと難しめかな。

▶なんと，このC-ON-Zというゲーム，LOGINのプログラムコンテストで入賞したものをタケルで売り出したものなわけだ。

で，このゲーム，シューティングゲームなわけだけど凄いのなんの。EASYでも1面目から敵がぼろぼろタマはいてくるわ，ボスキャラも弾幕のごとくタマは撃ってくる，レーザーは吹き出す，もうどうやってタマよけるんだかロシア人もびっくりっていうくらいの大騒ぎ（その代わり自機も3ウェイビシバシ，ミサイルも雨アラレに撃てるんだけど）。私ゃ，2面目までしか行けてません。

でも，いいんだ。BASICをコンパイルしただけで，アマチュアが作ってもあれだけのシューティングができることを証明したんだもの。難しくって先の面に進めなかろうがBGMがあるくせに効果音がなかろうが，X68000のマシンパワー＆X68000ユーザーのプログラミングパワーに乾杯だっっっ!!

熱中度▶▶▶▶▷▷▷　(で)

▶X68000お得意の横スクロール（もちろん多重）シューティングゲームC-ON-Zが登場しました。

ゲームは，やっぱりX68000だなあと思わせる派手な画面で，オプション兵器を最強にしたときなんかは敵の攻撃も重なって，超ハデな画面となります。そして雨アラレと降り注ぐ敵の攻撃をかわし続けていると……，やってきました敵の親玉！　こいつの攻撃も凄まじく，よけるだけでも本当にタイヘンです。ボスキャラは各面ごとにちゃんと攻撃パターンが違っているし，デザインのほうもよくできているのですが，デカいくせに当たり判定が小さいので倒すのにひと苦労，とっても疲れました。あとノリのいいBGMはあるのになぜか効果音がないんです。おかげでプレイしていたとき妙に寂しく感じました。

敵の攻撃パターン，ボスキャラ，そして背景と，全体的に見てもバランスのとれたなかなかのゲームでしょう。

熱中度▶▶▶▶▷▷▷　(純)

X68000用　5″2HD版　2,000円（税込）
ブラザー工業　☎052(824)2493

シューティングはX68000パワーだ！

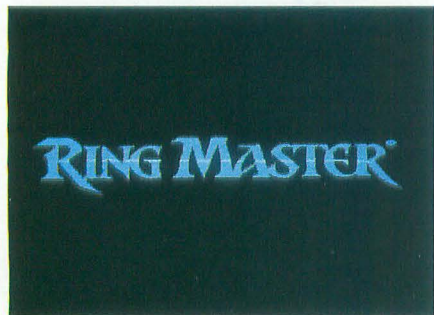
突然だけどX68000って本当にシューティングゲーム，多いですね。シューティングゲームがあるって簡単にいっちゃうけど，これは大変なことです。なにしろほかの種類（RPG etc.…）のゲームに比較しても短時間で動かすデータ量はダントツで多い。そんなわけでシューティングのプログラムを組むのはタイヘンなわけです。

にもかかわらず，アマチュアの作ったシューティングがまた出てきました，C-ON-Z。レベルはやっぱりプロの作品にはちょっと負けるけど（なにせあっちはプロ数人がかりで組むんだから），それでもきっちり遊べるシューティングを作っちゃうのは凄い。

でも，そのプログラムを組むパワーを与えてくれたのは，やっぱりX68000のマシンパワーだと思います。X68000でなら自分の思ったゲームがきっと作れると思い続けたからこそ，あの大変なシューテングのプログラムを作れたのじゃないかとも思うんですね，私は。

X68000のユーザーのパワーが凄いこともさることながら，ユーザーにそれだけのパワーをつけさせるような夢を見させるX68000というマシンは本当に凄い，と改めて思った今月の（で）でありました。

THE SOFTOUCH

●リングマスターⅠ
フィリアス・ノギスの暗雲

テーブルトーク感覚の本格派RPG

Kameda Masahiko
亀田 雅彦

時代背景や神話の構成をこと細かく設定し，なおかつキーボードからの入力によって登場人物と話すことができるというテーブルトーク感覚のゲーム。いままでのRPGでは物足りないという人には，なかなか新鮮でよいのでは……。

X68000用　5"2HD版3枚組　8,800円(税別)
ホビージャパン　☎03(354)9341

お待ちかねのリングマスターの第1章が，いよいよ発売になりました。そこで，不肖この私めがプレイする次第となりましたので，きっちりやらせていただきます。

さて，このリングマスターⅠ(今回は「フィリアス・ノギスの暗雲」の巻)，もとは98版からの移植ものでして，本家ではもうシナリオ2までリリースされている代物です。宣伝文句が，「テーブルトークRPGの雰囲気を，完全にコンピュータ上に再現！」ていうぐらいだから，さすがにみんな期待しちゃいますよね(注1)。X68000に移植するに当たっては，多少のヴァージョンアップが施されているようです。

フィリアス・ノギスの謎

「惑星アーサルドのゴドランダ大陸南端の国，フィリアス・ノギス。そこにはさまざまな事件が発生していて，リングナイト候補生である君は，それらに首を突っ込んでいくのであった。」という筋書は，どこにでも転がっていそうなお話です。ただ，このリングマスターが他のRPGと違うところは，そのフィリアス・ノギスの歴史まで細かく設定されていることです。ソフトのおまけのガイドブックには，神話の時代からの歴史が綿々と綴られています。これだけでも小説に匹敵するほどのものなので，プレイヤーを“その気”にさせるには十分です。

直接，本編のシナリオに関係ないところまで設定しておいたり，設定された歴史がシナリオに大きくかかわってくるのは，実はテーブルトークの常套手段です。プレイヤーがゲームの世界にのめり込まないと面白くないので，いろいろと架空の情報をでっち上げておくわけです。世の中には他にもたくさんのRPGがありますが，これほど背景世界を重視したゲームはないでしょう。ガイドブックに書いてあるとおり，RPGを楽しむにはまず雰囲気に浸ることが必要ですし，シナリオは大きくなればなるほど，テーブルトーク並みの設定が不可欠ですから，世界重視の傾向は大歓迎です。その意味で，リングマスターはコンピュータRPGというよりも，コンピュータ・テーブルトークと呼んだほうがいいかもしれません。

ちょっとほめすぎかな？　とも思えますが，新しくて面白いゲームを目指していることは，ほめるに値するでしょう。しかし実際のゲーム中では，残念ながらシナリオの素晴らしさがあまり伝わってきませんでした。まず町で情報を得てからダンジョンに繰り出して行って，そのクエストを解くというスタイルは従来のRPGと変わっていません。そうすることによって，最終目的へと近づいていく方法も同じです。それよりもいちばんいけないのは，最後の種明かしと最後の敵との闘いです。そこまで苦労に苦労を重ねてプレイしてきたのに，それを見た瞬間，脱力感に襲われます。詳しいことは書けませんが，こんな話は三流小説にもないでしょう。もっと勉強してほしいものです(注2)。これだけ書くのも，本当はこのゲームにそれだけ期待しているからです。最後を除けば，かなり面白いシナリオだと思うんですけどね。

本格的システム

ちょっと，コーヒーブレイク。テーブルトークの面白さは，実は充実したシナリオだけではありません。自由度の高さと，どんなことでもできてしまう柔軟性にあります(もちろんコンピュータRPGと比べた場合)。プレイしたことのない人にはわかりにくいかとは思いますが，システム側からもプレイヤー側からも，常識的なことはほとんどできるし反応もあるのです。なぜなら，結局は人間同士の会話から成り立つゲームだからです（おお！　まさにテーブルトー

おお，彼女こそは…

美しいグラフィックがその気にさせる

ク)。たとえば地面を掘る場合で，シャベルじゃなくても，剣とか手とかヘルメットでだって掘ることはできるはずです。そういうことにもすべて対応できます。

そこでリングマスターに話は戻ります。リングマスターのシステムで面白いのは，登場人物との会話ができることでしょう。単語を入力するとその情報をしゃべるくらいですが，アドベンチャー的な単語探しも面白いかもしれません（実際そんなに難しくないけど……)。町で出会う人にはしつこく聞いておいたほうがいいでしょう。酒場のおやじとか，一般市民はけっこう重要な情報源です。

次に，クエストの選択自由度も特徴です。始めと終わりはさすがにひとつですが，その間のクエストは，順番を変えたり省いたりしてもだいじょうぶなようです。もともとこのゲームは，自分のキャラクタを育てることよりもクエストを解くことに意味があるので，それもいいかもしれません。でもそうすると返すがえすも惜しいのは，最終目的の貧弱さなんです。いろんなクエストから得た情報の集大成として，デン！と構えていてもらえないものでしょうか……。それともうひとつ気になったのは，ダンジョンの広さです。広くしすぎてメモリ不足を招くよりも，狭くても探検していて楽しいダンジョンにしてほしかったです。

キャラクタの成長が目的じゃないといいましたが，パーティの個々の能力は，綿密に計算されています。精神面とか左手の能力とかもありますし，武器や防具に対する習熟度も設定されています。普通なら武器と防具ひとつずつですが，武器を両手にひとつずつ持ったりできて自由度は高いです。そして，習熟度が設定されている代わりに，このゲームにはレベルという概念がありません。より現実的な設定だといえますが，ろくに闘わないで最後まで行けるのには，拍子抜けしますよ。

プレイ中の画面

戦闘シーンではこれらすべてが考慮されるので，複雑な計算が必要となりますが，プレイヤー側はそんなことを気にすることなく闘えます（逃げることも大切なのですが……)。この点では，コンピュータがテーブルトークより勝っているようです。

実況プレイ

これだけじゃゲームの雰囲気が伝わらないと思うので，いちばん最初のクエストをお届けしようと思います。本当はこういうゲームの種明かしをするとつまらなくなるのですが，このクエストはいわば入門編なのでそのつもりで見てください。

我が友“るるぶ”は，ひとり宿屋の前に立っていました。まだ朝も早いのに，人々はもう活動を始めています。「おれはリングナイト候補生だ。なにか手柄を立ててリングナイトになるぞ！」と彼は解説しました。そこで町を歩くと，思わぬところで頼み事をされてしまったのです。渡りに舟とばかりに彼はOKしたのはいいけれど，なんとも貧弱な装備しかありません。「まあ，いいか」といって，町の近くの洞窟へ向かいました。

運の悪いことに，こうもりとの戦闘で彼の剣が折れてしまいました。しかも少々傷も負ったのです。「もう，帰ろうかな」と弱気になった瞬間，力強い仲間が現れたのでした。仲間を得た“るるぶ”は勇気百倍，狼だろうがトロウルだろうが闘いを挑みました。

洞窟の奥には地底湖がありましたが，泳ぎに自信のない彼らは湖をよけてもっと奥へと進むのでした。「あっ！　あれは墓だ！ドワーフの墓がこんなところに……」手を合わせると，どこからともなく声が。その方向には，なんとリングナイトの象徴“魂の剣”があったのです。“るるぶ”がそれを構えると，なにかしら正義の心が生まれて，フィリアス・ノギスを救おうなどとだいそれた計画を立ててしまったのです。

さらに奥には，このクエストのクライマックス，親玉トロウルと番人トロウルがいます。最後の力を振り絞って敵を倒すと，鍵を発見しました。どうやら2人目の仲間に出会えそうです。しかしながら，仲間といっても安心はできません。常に身の周りに気をつけましょう。とにかく，ここまでくればもう立派なリングナイト候補生です。

ゲームについてくるMAP

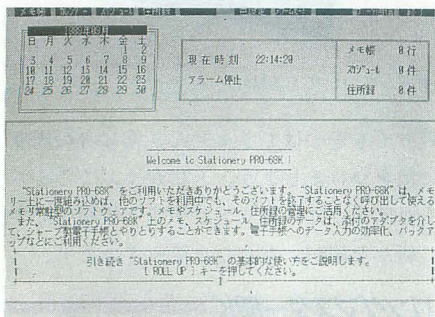
新たな旅が，あなたを待っています。

リングマスター68K

リングマスターは，細かい点を見れば行き届いた設計になっています。朝，昼，夕，夜と画面が変わりますし，主人公の装備は常に表示しているので一目瞭然です。また，マウス操作も無難にこなしているといえます。しかし，シナリオ全体を通して見たときには，貧弱さが目立ってしまいます。グラフィックとか，各クエストのダンジョンに相当のメモリを浪費しているようなので，もっとテーブルトーク風の楽しさを味わわせてほしいものです。そうはいっても，いままでの RPG よりはテーブルトークに近いものですから，確実に期待の持てるゲームではないかと思います。

ゲームとは直接関係ありませんが，このソフトのプロテクトは変わっていて，ゲームディスクのバックアップが取れるようになっています。その代わり，1冊の本に書いてある4桁の数字を，ゲームの始めと終わりに照合するようになっています。これならディスクの不慮の事故も防げるので，安心ですね。こういう方面の研究も続けてもらいたいものです。

注1：テーブルトークとは，コンピュータRPGのもとになったもの。雰囲気に浸り切ることができれば相当面白い。いわゆるオタク受けのするゲーム。やってる奴はかなりあぶないと見ていい。
注2：本当は最後に「第2章へ続く」というメッセージが出たので，あれは単なるプロローグだといわれればそれまでだが，それにしては8,800円（＋消費税）は高すぎる。それに，エンディングもやけに簡単だったし……。

●Stationery PRO-68K

X68000に常駐するプライベートツール

Ogikubo Kei
荻窪 圭

X68000らしい贅沢なメモリ常駐型ソフト。日常生活を取り巻くプライベートなデータを他のアプリケーションを使いながらいつでも参照できます。念願の電子手帳とのデータ交換も可能になりました。

X68000用　5"2HD版　14,800円（税別）
シャープ　☎03(260)1161
＊シャープ電子手帳との接続には通信ケーブルCE-200（別売り）が必要です。

サラリーマンの巣窟である会社には，灰色の机が所狭しと並んでいて，そのうち何割かの天板の上に“黒い色をしたデジタル時計とカレンダーとペンとメモ帳がひとつにくっついた机上文具”が乗っかっている。このStationery PRO-68Kというやつは，X68000のHuman68k上に乗っかった“黒い色をしたデジタル時計とカレンダーとペンとメモ帳がひとつにくっついた机上文具”なのである。おわかりかな？

と，これで終わってもいいのだけれど，それではミもフタもない。ここではStationery PRO-68Kを2つの最重要ポイントから攻めていこう。ひとつは，いつでも手を伸ばせば使える机上文具的便利さを実現した“メモリ常駐型プログラム”という点，もうひとつは机上文具の持つ“メモを破って手帳に挟んで持ち歩く”的機動性を実現した“電子手帳との通信機能”だ。

Stationeryは常駐する

で，Stationery PRO-68Kの使い方だが，STPRO.Xというプログラムを実行すればよい。すると，実行したよん，というメッセージを残して，メモリに棲みつく。このままでは単に常駐しただけだから，ただフリーメモリが減っただけでおいしくもなんともない。常駐したStationery PROを呼んでやらねばならないわけだ。

Stationery PROはどういう条件のときに活性化して割り込んでくるかというと，OPT.1キー＋UNDOキーを押したときである。これで写真（左）のようなメニューが出る。ここでファンクションキーを使ってメモ帳やらカレンダーやらスケジュールやら住所録を呼べる。が，メインメニューを介さずとも，いきなりOPT.1＋F1とかOPT.1＋F4とかでいきなり目的の機能を呼ぶこともできる。ファンクションキーよりOPT.1に近いUNDOのほうが押しやすいので，たいていの場合メインメニューを起動してしまうけどね。

●利用の条件

だがしかし，この世の中，そんなおいしい話はないのである。いつでもどこでもキー一発で呼び出せて，なおかつ電話帳したりカレンダーしたりという便利なやつがそうそういるものではない。Stationery PROでも，そうだ。いくら常駐していても，そいつを呼び出すための制約というものがある。

まず，呼び出すときの画面が96×32行モードであることだ。グラフィック画面でいえば，768×512ドットの場合しかだめなのである。だから，MUSIC PROやZ's STAFFを使っているとき，512×512ドットモードでゲームをしているときは使えないのである。ここで，いつもゲームしかしないユーザーは脱落なのだ（こらこら本を閉じないように）。

さらに，使っているソフトがマウス専用でキー入力を禁止していたりしてもだめなのである。たとえば，ビジュアルシェルだ。ビジュアルシェルからだと，キー入力可能な画面（たとえば，ノート）からしか呼べないのだ。

そのうえ，常駐するとメモリを約300Kバイトも使うので，メモリを増設していないと，Stationery PROのおかげで起動できなくなるソフトが多い。そこでマニュアルに載っている動作確認ソフト（シャープ製のみ）をお載せしよう（表1）。

しかし，である。制限が多くても，便利なものは便利なのだ。ついさっき，友人のところへ電話する約束があったのを思い出したわけだが，わざわざ手帳を広げるまでもなく，この原稿を書いていたワープロ上でちょいとOPT.1＋F4キーで住所録を呼び出して調べるだけですんだ(実話)。うまく使えばこのように便利なのだ。

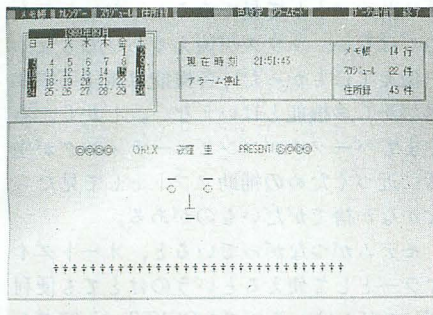
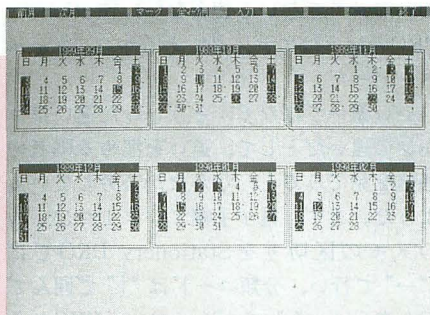
●環境ファイル

話を戻す。メインメニューには今月のカレンダー，現在時刻，メッセージウィンドウ，アラームモードなどが現れる。メッセージウィンドウにはあらかじめ指定しておいたテキストファイルが表示されるうえ，上下にページスクロールしてくれるので，アイデアしだいではヘルプ機能的に使えるだろう。ちなみに，サンプルでは簡単なStationery PROのヘルプがついてくる。

このヘルプファイルも，住所録ファイルもメモ帳ファイルも，とにかくStationery PROではファイルを扱うわけだが，自分で好きなファイル名を指定できる。環境ファイルに“メモ帳はこいつを使うよ”などと

表1　同時に実行可能なアプリケーション

Communication PRO-68K
CARD PRO-68K*
DATA PRO-68K*
BUSINESS PRO-68K*
C compiler PRO-68K*
NEW PrintShop PRO-68K*
TOP 給与計算エキスパート*
TOP 財務会計*
THE 福袋 V2.0
AI-68K(Staff LISP / OPS PRO-68K)*
〈本体同梱〉
日本語ワードプロセッサ*
X-BASIC
ED.X(スクリーンエディタ)

＊要2Mバイト

荻窪氏の作ったメニューファイル

カレンダー

書いておけばいいわけだ。また環境ファイルといえども，OPT.1＋UNDOで呼ぶたびにアクセスしていてはたまらないので，Stationery PROを常駐させる場合にのみアクセスに行けばよい。

ついでに言っておくが，常駐させるときは環境ファイルだけでなく，全データファイルを常駐させてしまうのだ！　おかげでメモリをたくさん食うわけだが，Stationery PROで住所録やメモ帳を書き換えてセーブしなかった場合でも，常駐を解除したり電源を切ったりしない限り，メモリ上にいるわけだから，OPT.1＋UNDOで呼び出す限り保持されている。保存したくない情報はセーブさえしなければ常駐解除で消えてくれる。うーん，さすが。

なお，環境ファイルを書き換えたら，一度常駐を解除して（もう一度STPRO.Xを実行するか，STOFF.Xを実行する），再度常駐させないと，意味はない。

また，環境ファイルには画面の色設定とか印刷時の１ページ当たりの行数とかモデムの設定などなども書ける。さらに，起動時オプションで各データファイルの大きさ（デフォルト16Kバイト）も指定できる。

電子手帳と仲好しとは

君は，あの景山民夫が宣伝している電子手帳を持っているか。カシオやキヤノンじゃなく，シャープのやつだ。あれと，X68000でデータのやり取りができるのだ。

●電子手帳と接続する

Stationery PROの上位コンパチ（？）である電子手帳をX68000につなぐには，ジョイスティック端子を使う。おお，偉大なりジョイスティック端子。桒野氏が万能赤外線リモコンをつないだと思ったら，今度は電子手帳とつながるのだ。サードパーティからRS-232Cを介してつなぐセットも出ているらしいが，やはりX68000であるからには，でかくて背面にあるRS-232Cよりも小さくて前面のジョイスティック端子である。

ジョイスティック端子にはStationery PRO-68K付属のインタフェイスをつなぐ。そして別売りの電子手帳用ケーブルを差し込んで電子手帳とつなぐのだが，ここでシャープに文句を言っていいだろうか。実はケーブルが短い！　のである。電子手帳同士ならともかく，電子手帳とパソコンとなると，36cmではいささか短い。せめて50cmくらいは欲しい気がする。

●転送をしてみる

で，両者をつなぐ。転送できるのはメモデータ，スケジュールデータ，住所録データの３つである。

転送モードには個別転送と全データ転送の２種類がある。全データ転送というのは，３種類のデータを全部転送してしまえというやつで，転送先のデータは消え，新しいデータが代わりに入る。個別転送では，１つひとつ，たとえば住所録であればひとりずつ転送するモードで，転送先のファイルに追加されていく。

この機能によって，普段はX68000で住所録管理をしておいて，必要なものだけを電子手帳に移して使うとか，外出先で新しく入手した情報をX68000に転送するとか，いろいろと便利だ。普通，パソコンに入っているデータは汎用性や蓄積度は大きくても機動性に欠ける。電子手帳のデータは機動性に富んでいても汎用性や編集能力に欠ける。両者の合体は好ましいことだ。もっといいのは小さくて安いラップトップ（頑張れ，ダイナブック！）だけどね。

さて，データ転送は簡単。とっても簡単だ。住所録のひとりを転送する例で説明しよう。まず，Stationery PROを住所録モードにし，電子手帳を電話キーを押して住所録モードにする。

ここで，Stationery PROから電子手帳を考えよう。そのときは，送りたいデータのところへカーソルを持っていく。

続いて，電子手帳の機能キーを押し，オプションキーを押す。すると，電子手帳が“受信モードです”となるから，X68000ではF9キーを押す。転送される。終わり。

逆に電子手帳からX68000の場合は，電子

常駐プログラムとは

Oh!Xにも常駐プログラムはいくつか掲載されたことがあるが，常駐の概念については大きく取り上げられたことがなかった気がするので，その話をしておこう。常駐というのは，メモリ上の空いているところへ居座ってしまって，帰らないことをいう。

何がいいかというと，１回常駐させてしまえば，いつでもすぐ呼び出せる（あるいは，勝手に働く）ことである。チャイルドプロセスを使うでもなく，コマンドを入力する必要もなく，いつでも“居る”のだ。逆に何がよくないかというと，メモリを食ってフリーエリアが小さくなって地価が高騰することである。

たとえば，よくある常駐プログラムに“勝手に動くやつら”というのがある。常駐時計もその一種だ。時計の場合はいつでも画面の一部を陣取って時刻を教えてくれるのである。もちろん，メモリの一部を占領するだけでは時刻の表示なんてできない。何をしているかというと，一定時間ごとに（しかも，目に見えないほど頻繁に）“俺は画面の隅に時刻を表示しにいくぞー”と叫んで，CPUの68000君が何をしていようが構わず“おらおらおらおらー”と割り込んで画面に時間を表示してしまうのである。

昔，I/O誌に載ったシーモンキーというのもあった。これは何かというと，画面中を数匹のシーモンキーが泳ぎ回るだけである。これもユーザーの目に見えないくらい素早く割り込んできて68000のパワーを掠め取って動くので，はなはだやっかいなやつであった。だから，エディタを使っていようが，BASICを使っていようが，かまわずにシーモンキーは泳ぐのである。

かの98ではカーソルをネコが追いかける常駐プログラムなんかが有名だ。

もう少しタチの悪いのになると，“ある一定の条件が満たされたとき急に割り込んでくるやつ”がある。たとえばHuman68kのVer.2.0に付いてくる“TIMER”コマンドなどもその一種である。設定時間になるまでじっと陰に隠れていて，突如グアっと活動を始めるのである。立ち上げて２時間以上立つと「私，疲れちゃったあ」と叫ぶようにするのだって，簡単だ。

時限爆弾でなくともよい。電脳倶楽部に昔載ったやつに，ディスクを出し入れするたびに指定したPCMファイルを鳴らす常駐プログラムがあった。

このように，常駐プログラムというのは，普段は隠れていて，何かあるとしゃしゃり出てくる子悪魔のような存在なのだ。隠れているやつがいるかどうか心配なときは“PROCESS”コマンドを実行するといい。メインメモリ上にいるやつを教えてくれるスパイ衛星なのだ。

もちろん，プログラムを常駐させるには，常駐ボタンを押して起動すればよいなどという都合のいい話はないわけで，それなりに常駐して働いてくれるように作らねばならない。具体的に何をどうすればそういうことができるのか，という技術的な話は調べるなり，詳しい人に聞いたりするように。Oh!Xに強く要望すれば，そのうち誰かが書いてくれるかもしれない。

手帳の送りたいデータを呼び出してから，先に X68000 の F9 キーを押す。すると，X68000が受信モードになるので，電子手帳で機能キーを押し，オプションキーを押す。転送される。終わり。

つまり，どちらを先にデータ転送モードにするかで送受信が決定されるのだ。先にデータ転送モードにしたほうが，受信となる。うーん，怖いほど簡単だ。

●電子手帳とのソフトの違い

ただ，ここで気になるのは，さっきもちらりと書いたけど，なんかStationery PROが電子手帳のサブセットみたいなのだ。もちろん，データ量や画面の情報量(間違っても電子手帳で住所録の一覧なんて見られない）なんかではパソコンが上だけれど，電子手帳だと2行にわたって名前を入れたり(名前と会社名とか)，2行にわたって電話番号を入れたりできる（自宅と会社とか，会社とそのFAXとか)のに，Stationery PROだと1行しかスペースがないので電子手帳から2行のデータを送ると，

“荻窪圭↓Oh!X”

となってしまうのだ。うーん，どうすればいいかと聞かれても困るけれど，もう少し柔軟な対応が欲しかった。

それから，メモ帳というのは元来“小さく切った紙が綴じてあるもの”であって，決して“馬鹿長いロールペーパー”ではない。電子手帳のメモ機能は，分類コードまであっていかにもメモ帳だが（あんな小さな画面ではどだい大きなファイルでなく細切れのメモでないと使えない)，Stationery PROのほうはエディタであるから，ロールペーパーだ。

いい悪いではなく，そういった違いはマシンの違いだとして，違いを埋める両者の通信を見てみると，なかなか苦労しているようだ。まず，電子手帳でいう1つひとつのメモの区切りをStationery PROでは“→”で行い，分類コードは“{”で囲んで示している。そして，Stationery PROから電子手帳へのメモの転送は区切りコードを1単位として行われる。

また，電子手帳にはX68000の持っていない記号や絵文字があるが，X68000側は外字で記号のみ対応している（ということは，Stationery PROに付いてくる外字データを使わないと変な文字が出てきたりする)。絵記号は対応していない。

こんな風に両者は違うわけだが，電子手帳は機動性はあるけれども，その編集機能はお世辞にも（パソコンに比べると）使いやすいとはいえない。Stationery PROはその点は押さえているわけだ。

外出先ではとにかく（間違えようがなんだろうが）どんどんメモし，あとでX68000で綺麗にして戻すというのがいちばん賢いのではないだろうか。

電子手帳とのデータ通信

第3部　その他の機能とは

常駐性と電子手帳の話ばかりになってしまったが，まあ，Stationery PROを“常駐”や“電子手帳との接続”と切り離してユーティリティとして見てみると，たいしたものではない。マウスが使えるわけでもなく（使えてほしかった)，各機能は電子手帳準拠だから多機能とはいえない。しかし，あくまでパーソナルコンピューティングが生活に近づくための補助ソフトとして見たら，なかなか捨てがたいものがある。

モデムがつながっていると，オートダイヤラーとして使えるというのはとても便利だ。モデムはヘイズでもCCITT V.25でもよい。また，電子手帳用のハンディプリンタ(CE-60P)も使えたりするし，付属のファイルコンバータで，CARD PROや普通のテキストファイルへも変換できる。

では，詳しく触れられなかった基本機能について流しておこう。

●メモ帳だ

メモ帳は92行横スクロールなし（つまり，右端まで行ったら折り返す）簡易エディタだ。簡易とはいえ，CTRLキーを使った編集機能やファンクションキーを使った検索/置換もあって，普通に使える。大きなファイルを扱おうとしない限りパフォーマンスは悪くない。スクロールは遅いけど，ページスクロールは速い。

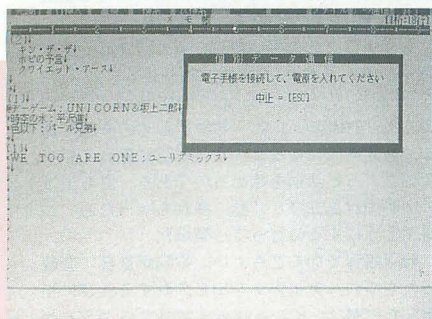
もちろん，ファイルの読み書きはできるから，別のプログラムを使用中に“ちょっと，あのファイルの中身を見ておきたい”なんてときに便利だ。

使い方は二者択一といえるだろう。X68000使用時のメモやファイル編集用に使うか，電子手帳とのリンクを第一に使うかである。なお，印刷機能もある。

●カレンダーだ

カレンダーである。メインメニューではその日のカレンダーが表示されるだけだが，カレンダーモードになると，写真のように半年分並んでくれる。上段中央が現在の月だなんてのがいい。1カ月前まで同時に見られるのだから。当然，ずりずりと表示月を変えることもできる。

F4キーにマークという機能があるのが気になる。これは，カレンダーに休日をマークする機能だ。マークした日は色が変わって反転する。つまり，元の状態ではまっさら。日曜だろうが祭日だろうがわからないのである。実は，これは電子手帳のカレンダーも一緒だ。マークは256個まで設定でき，ファイルに格納されるので，私は片っ端から休める日を（土曜も含めて）マークしてある。

適当な日にカーソルを合わせてF6キーを押すと，スケジューラに入ってスケジュール入力モードとなる。カレンダーの日付

電子システム手帳 PA-8500

ここで，Stationery PROの仕様に大きな影響を与えた電子手帳の機能を語らずに話は始められない。

今回使ったのはPA-8500という最新の機種だが見かけも悪くない。この電子手帳はデフォルトで（つまり，別売りICカードを入れなくても）7つの機能を持っている。

カレンダー，スケジュール，電話帳，メモ帳，計算，世界時計，時計だ。あろうことか，パソコンのStationery PROより2つも機能が多いではないか。Stationery PROは電子手帳のサブセットだったのか。Stationery PROは電子手帳の下位コンパチに過ぎないのか！

それはそれとして，電子手帳であるが，これがなかなか面白い。操作法などかなりクセがあって慣れが必要だが，あの大きさでこの使い勝手ならば，金余りビジネスマンに売れるのもわからないでもない。

の右側にある小さい点はその日のスケジュールの有無を表しており，上の点が午前の，下の点が午後のスケジュール印である。時刻指定をしないと，午前にマークされる。このあたりは，電子手帳とコンパチだ。

●スケジューラ，だ

スケジューラは日付，時刻，内容のリストがずらっと並んだ画面になっている。ここにいろいろと書き込むのである。書き込める内容はもちろん，電子手帳コンパチ。スケジュールが多い人のために，検索や印刷機能もある。

メインメニューでアラーム待機モードにすると，いちばん近いスケジュール時間に達したとき，ビープ音が鳴る。よくやるのが，“24：00　ペヤングソース焼きそば出来上がり”というやつだ。

なお，このアラーム機能は電子手帳より劣る。最大の難点は，スケジュールした時刻にしかアラームがならないことだ。当たり前かと思うかもしれないが，電子手帳では，セットした時間の５分前や10分前に（なんと，60分前まで設定可能）鳴らすことができるのだ。たいていの場合は早めに鳴ってほしいと，思うだろう。

●住所録もあるでよ

住所録もスケジューラ同様，名前と分類と電話番号と住所のリストの形で表示される。表示フィールドより長い文字列を入れても，きちんと保持していてくれるから表示が短くても大丈夫。

例によって，検索や印刷機能もあるほか，データソートも任意の項目でできる。データベースとはいえないけれど，立派なアドレス帳ではあるし，他のアプリケーションのデータと互換性があるから便利。

たとえば，私はKamikazeのデータベースに住所録を作り，それをカルクシートへ読んでリフィルサイズに調節して印刷してシステム手帳に入れて使っているが，Kamikazeの標準テキスト出力（CSV形式）でセーブすればそのまま読めるので，Kamikaze上からStationery PROを呼んで標準テキスト出力されたファイルを読み，必要なものを電子手帳に転送するという簡単な方法でできるようになった。

ただ，このご時勢。電話番号や住所は２種類までサポートしてほしい。実家と下宿とか，自宅と会社とか，そういったケースは多いはずだ。

我が家でもっともStationery PRO-68Kが威力を発揮するのは，電話をかけたりかかってきたりしたときである。電話中というのはカレンダーを見たりメモをしたりス

住所録

スケジュール

ケジュールを調べたりが非常に多いのだが，わざわざ乱雑な部屋からメモ帳を捜したり（そして，後でその紙をなくしたりする），壁のカレンダーをめくって次の月を覗いたり，手帳を引っ張り出して次の予定を調べたりはしたくない。こんなとき，キー一発ですんでしまう。座ったままで，だ。

＊

Stationery PROは，パソコンを生活に役立つ道具として使いたいと思っているごく一般の人が相手だ。だから，Stationery PROで入力したデータをコピーバッファなどを介して使いたいとか，辞書メンテナンスツールが使えるようにしてほしいとか，もっとエディタとして使いやすいものにしてほしいとか，仕様をユーザーに開放して機能を追加できるようにしてほしいなどというのはとりあえず，当たらない。

ただ，電卓機能がないのだけは痛い。Stationery PRO上にいると，OPT.1＋OPT.2の簡易電卓さえ使えないのだ。関数電卓とか，10進－16進変換なんていわないから，普通の電卓くらいつけるのが机上文具の使命だと思う。

毒を食らわば皿までつきあって

今回はそういったわけで，電子手帳コンセプトだったわけであるからこれでいいとして，問題は次回である。常駐させていつでも呼べるというのはとんでもない魔力である。メモリが少々少なくなったって，その分RAMディスクが削られたって，いいのである。だから，ビジネスユーザーや電子手帳ユーザーだけでなく，もっと普通のパソコンファンが喜ぶようなものを次にはお願いしたい。

幻の史上最強８ビットパソコンMZ-2500にはアルゴ機能というのがあった。アルゴ機能とは？　立ち上げ時にメモリに常駐し，電卓・カレンダー・カラーシミュレーション・エディタ・オートダイヤラー（住所録）・ディレクトリサーチツールと豊富なラインアップがあり，もちろん2500の256Kバイトの RAMでは全部を常駐させるのは無理で，任意のものだけを選んで組み込めるようになっていた。BASICでプログラムを作りながらアルゴエディタでその原稿を書くといった荒技もできたのだ。

あのアルゴ機能は，以下の点でStationery PROより優れている。

・電卓の計算結果やサーチしたファイル名をコマンドライン（呼び出した画面のカーソル位置）に返せる。

・電卓やカレンダーは元の画面にオーバーラップして表示される。

・任意の機能を選択して組み込める。

・ユーザーが機能を追加できる（本誌でもアルゴゲームが掲載された。アルゴ機能を使って，いつでもできるゲームだ）。

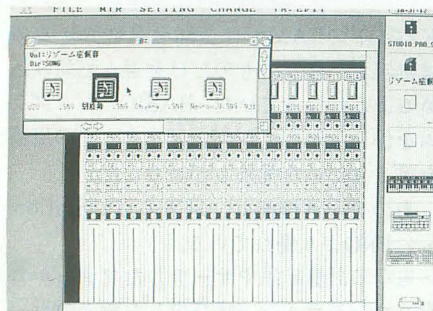
・基本的に，どの画面モードでも使える。

これらの機能を踏まえ，次には“お助けPRO-68K”や“いつでもどこでも誰とでもPRO-68K”と称して，便利なツールを提供してもらいたいものである。

そのときはぜひ，関数電卓やコピーバッファなどによる元の画面とのデータのやり取り，マウスのサポート，512×512の画面からでも呼べる柔軟性，プログラムのモジュール化（組み込む機能の選択），辞書メンテナンス，ファイル関係のユーティリティなどなどもつけてもらいたい。

究極のデスクトップパソコンは“ユーザーが家にいる時間は常に電源がオンの状態にしておきたい”くらいのものでなければならない。今回のStationery PRO-68Kも「ちょっとメモを」とか，「ちょっと，電話を」というときにすぐ使える仕様がいいわけだが，電話番号を調べたり手元にメモ帳がないときにわざわざX68000に電源を入れるのはバカバカしい。電子手帳とつなぎたいだけなら常駐にする必要はなかったわけだから，そういった究極のデスクトップパソコンを目指しているのだろう。

単なる一介の個人ユーザーに過ぎない私も，そんな，いつでも電源を入れておけるパソコンに育ってほしいと常々願っている。

●Musicstudio PRO-68K用ソングファイル

自由な音楽を目指す人のためのメディア

Deguchi Kaori　Ogikubo Kei
出口　香・荻窪　圭

プロのミュージシャンが作った曲をアレンジして楽しめる，ということで話題になった新感覚のツール，Musicstudioデータ曲集。そのシリーズに新しく2本がリリースされた。音楽を聴くのが好きな人にはうってつけのソフトだ。

X68000用　　5"2HD版 各5,800円(税別)
(要Musicstudio PRO-68K，MIDIボード&MT-32)
・佐藤允彦／リゾーム症候群（全8曲）
・関根安里／スケッチ（全8曲）
サン・ミュージカル・サービス
☎03(419)8839

今や作曲もコンピュータの時代。X68000もMIDIボードやMusicstudio PRO-68Kが出現したおかげでようやく本格的にパソコンミュージックができるようになりました。でも，それはあくまでオレはコレで作曲するんだ，って人や，私はTMネットワークの曲をコンピュータで再生したいわ，などという意気込みを持つ人たちのためのようなものでした。そんなわけで登場したのがこのMusicstudio PRO-68K用ソングファイルシリーズです。このシリーズはプロのミュージシャンが創った曲を楽しむために作られたデータ曲集で，ただぼーっと聴くのもよし，自分流に思いっきりアレンジするもまた楽しといったソフトなのです。今回は前4作に続きリリースされた2作品を紹介します。

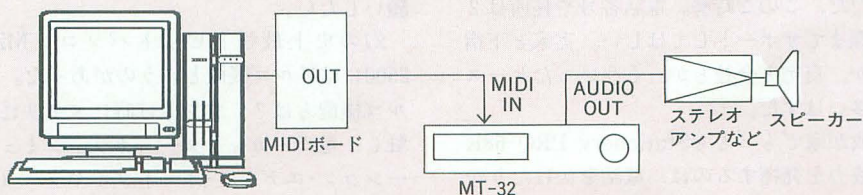
が，その前に知らない人もいるかと思うので一応接続方法を説明しましょう。まずX68000にMIDIボードCZ-6BM1を装着します。そしてそのボードにMT-32(ローランド)をMIDIケーブルでつなぎます(ボードのMIDI OUTとMT-32のMIDI INとをです)。さらにアレンジを楽しみたい人は，ボードのMIDI INとMT-32のMIDI OUTをつないでください。あとは，MT-32のオーディオ OUTからステレオなどのオーディオ INへつなぐだけです。

リゾーム症候群

作曲者はジャズ・ピアニストの佐藤允彦。ジャズが好きな人なら一度は聞いたことがあるでしょう。ということで曲の批評は本誌でおなじみの荻窪圭氏にしていただくことにしました。

1曲目――暗い部屋でじっと聴くと脳髄をかきむしり，ミニマルなリピートがサブリミナルな領域に警告を発する。

2曲目――パーカッションの音があまりにもMT-32だけれども，ポップになることを拒否している。脱力エスニック。

4曲目――実験音楽のよう。サイバーパンクエスニックと呼んであげよう。

6曲目――小国客死の村に着いたはいいけど，ここはどこ？　私は誰？

8曲目――インディアなエスニック。中近東の山奥で密教が舞踏している心地。

総評――人生に疲れたときに聴いてはいけない。エナジードレインミュージックだ。ミニマルミュージックとクラフトワークをほうふつとさせるテクノなリピートとエスニックを混合したような感覚がかきむしる。今ひとつTV的な曲がなくて残念。

スケッチ

作曲者は関根安里。「銀河漂流バイファム」の歌を唄ってたTAOというバンドの人と言えばわかりやすいかな。では再び荻窪氏に批評していただきましょう。

2曲目――007はいらない。

4曲目は……あっと，荻窪氏が嫌いなフュージョンだ。ちょっとあっちいってて。

(荻窪：フュージョン嫌い。ブツブツ……)

出口：テクノフュージョンとしての出来は悪くないかな。シャカタクっぽいけど派手だし，元気があっていいと思います。

5曲目（荻窪氏ブツブツ言いながら戻ってきました）――大仰な曲って好き。フェイドアウトでごまかすなよな（こらこら)。

6曲目――音を重ねるのはうまいけど，工夫が……（きれいだと思うけどな)。

総評――プロの職人的な欠点が目立つ。バイファムの主題歌は良かったのに。が，グレードは低くない。聴きやすくまとまっている。

ということで偏見に満ちた批評を荻窪氏にしていただきました。私の感想としては2作品ともやっぱりプロだな，と思うほどエフェクタの使い方が上手です。耳触りになりがちな効果音をさらっと聴かせる，これがプロの手法なんですね。ということで今回はこの辺で終わりにしたいと思います。

●発売中のソングファイルシリーズ
知恵ある暮らしの味／国元佳宏
インセクト／佐久間正英
ピーセス・オブ・ワーク／本多俊之
あの娘のDNA／戸田誠二

図1 MIDIシステムの構成

画面スクロールの手法

Izumi Daisuke 泉 大介

今月は，読者の要望が多い画面スクロールの実習です。テキストとグラフィック画面のスクロールからグラフィック制御の基本までを解説します。プログラム自体は簡単ですのでしっかりと解読して仕組みを理解してください。

前回はX68000の素晴らしいグラフィック機能のサンプルとして，チェス盤を表示するプログラムをお届けしました。本格的な(?)マウスオペレーションのチェス盤ですが，使って頂けたでしょうか。[0] 今回はいまやアクション型ゲームでは常識となった画面のスクロールについて解説したいと思います。

画面スクロールの原理

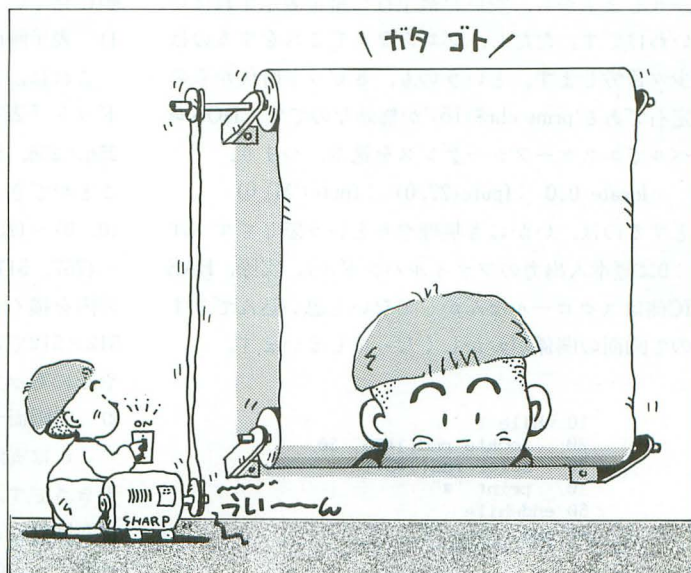
先月，ゼビウスに触れたときの驚きやスペースハリアーを見たときの感動をお話しし，1つひとつステップアップしていこうと書いたところ，「スクロールには興味があるのでぜひ早く解説してほしい」というお便りが何通も届きました。本当はもう少したってから行うつもりだったのですが，ご要望にお応えして今回は画面のスクロールを取り上げます。これからも皆さんのご要望に沿って進めていくつもりですので，こんなことをやってほしいというご希望がありましたら，編集部にどんどんお便りください。では本題に入りましょう。

スクロールというのは本来「巻物」という意味です。そう，忍者が口にくわえてドロンとやるあれですね。巻物には2つの軸があり，一方の軸に巻き付けてある紙を他方の軸に巻き取りながら読んでいくようになっています。また，逆に巻き取れば，以前読んだところをもう一度読むこともできます。これをコンピュータの画面に当てはめ，コンピュータの画面がイラストのようになっていると考えてみてください。画面の上の軸に巻き取れば表示されている文字は上に流れ，下の軸に巻き取れば下に流れます。コンピュータの世界では巻き取るときに文字が画面を流れる様子をスクロールと呼んでいます。文字が上に流れれば「上スクロール」，下に流れれば「下スクロール」です。

X-BASICで適当な命令をどんどん入力していけば，やがて画面に入りきらなくなり，新たに文字を入力するたびに表示されている文字が上に流れていきますね。そう，上スクロールしているのです。逆に，下にスクロールさせるにはどうすればいいでしょうか。これにはカーソル移動キーの上にある6個のキーを使います。まず「HOME」と書いてあるキーを押してみてください。カーソルが画面左上に移動しますね。次に「ROLL DOWN」と書いてあるキーを押してみてください。どうですか，1行下に移動しましたね。このキーは，プログラムを作っている最中も重宝します。プログラムの途中に新しい行を追加したくなったときには，プログラムを追加するところへカーソルを移動し「ROLL DOWN」キーを押せば，そこに1行分の空行を挿入できるのです。追加する場所にプログラムを書き込めるわけですから，確認しやすく，行番号の付け間違いなどの防止にもなります。

テキスト画面のスクロール

では，星をスクロールさせる簡単なプログラムを使って，実際にどのようにすればスクロールさせることができるのか実験してみましょう。画面の最下

0) 先月のプログラムではキングとクイーンの位置が反対でした。某市販ソフトを参考にしたのですが……。

行で命令を入力すれば，画面が上にスクロールしますね。X-BASICでは(そして，大抵のマシンでは)画面の最下行で改行すると，画面がスクロールするようになっているのです。ですから，

```
10  for i=0 to 3000
20      print i
30  next
```

というプログラムを実行すれば，改行しながら数を順に表示していき，画面の最下行に達したところからはスクロールしながら表示されることになります。20行目のprint命令が“;”で終わっていないことに注意してください[1]。

1)「print i;」ならば，数字を表示したあと改行せずに次の文字を表示しますので，このプログラムでは数字が羅列されることになります。

もっと凝って，次のようなことをすることもできます。まずはこのとおり入力して実行してみてください（リスト1）。

どうです，なかなか綺麗なものでしょう。locateはこれから文字を表示しようとする座標を教える命令です。画面最下行の座標は（～,30)になります。そこでx座標を適当にとりながら，適当な色をつけて“*”を表示すれば，色とりどりの星が流れていくように見えるという仕掛けです。新たに登場したrnd関数は，0≦x＜1の適当な小数（実数型の乱数）を返す関数です。1になることはありませんから，rnd（ ）*96は0以上96未満の小数になります。さらに，locate命令に小数で座標が与えられると，小数部分が切り捨てた整数として扱われますので，x座標は0以上95以下となり，表示できる範囲を越えることはありません。30行も20行と同様にこれから表示する文字の色（1～3）を指定しています。

●その他の方向へのスクロール

画面最下行で改行するのと同じ発想で，文字を下スクロールさせることもできるはずです。つまり，画面最上行でROLL DOWNキーを押せば下スクロールしますから，空いた最上行に星を表示すればいいわけです。ただし，プログラムでこれをするのは少々苦労します。というのも，8ビット時代からの定石である‘print chr$(15)’が無効なのです。DOSレベルでエスケープシーケンスを送る，つまり，

```
locate 0,0  :fputc(27,0)  :fputc('M',0)
```

とするのは，いかにも無理やりという感じです（注:0は標準入出力のファイルハンドル)。実際，BASIC側はスクロールなんかしてないと思い込んでますので画面の関係がおかしくなってしまいます。

リスト1　星のスクロール

```
10 while 1
20   locate rnd()*96, 30
30   color rnd()*3+1
40   print "*"
50 endwhile
60 end
```

グラフィック画面のスクロール

従来のマシンでは，グラフィック画面をスクロールさせるのは大変な作業でした。考え方は文字を表示しているテキスト画面のスクロールと同じなのですが，グラフィック画面は画面の最下段にドットを表示したからといって自動的に1ラインスクロールしてはくれません。そこで現在画面に表示されている絵を1ライン上に「自分で」転送する必要があります。このためにはマシンのハードウェアを研究しなければならず，実現するにはマシン語の力を借りなければなりませんでした（わかってしまえば転送の原理は実に簡単なのですが)。

X68000でもこのような面倒な手順を踏まなければならないのでしょうか。いいえ，X68000のハードウェアはスクロールを強力にサポートしてくれるため，簡単に実現させることができるのです。ではそのための予備知識として，グラフィック画面の構成をもう少し突っ込んで調べてみましょう。

●X68000のグラフィック画面構成

X-BASICでグラフィック画面の初期化関係の処理を一手に引き受けているのはscreen命令です。ではX-BASICのマニュアルで，screen 命令を見ながらグラフィック画面の構成を説明していきましょう。

screen命令は4つのパラメータをとります。最後のパラメータは，グラフィックを表示するかどうかを選択するものですから簡単ですね。また3番目のパラメータはディスプレイが高解像度かどうかを選択するものです。つまり512ライン表示できるか，表示できないかということですね。1番目のパラメータは表示画面サイズの指定で，2番目のパラメータは実画面サイズおよび表示色の指定です。1番目から順に見ていきましょう。

1）　表示画面サイズ

これは，グラフィックを画面の縦横にそれぞれ何ドット「表示」するかを指示するパラメータです。256×256，512×512，768×512の3つから選択することができます。画面左上～右下の座標はそれぞれ，(0, 0)～(255, 255)，(0, 0)～(511, 511)，(0, 0)～(767, 511)になります。座標(256, 256)に半径250の円を描くと，256×256の画面では円の左上1/4が，512×512では画面いっぱいの円が，768×512ではやや左に寄って円が表示されることになります。

2）　実画面サイズおよび表示色

これは実際にグラフィックを「書き込む」範囲の大きさです。実画面サイズと表示できる色数には密接な関係があります。最も大きいのは1024×1024の画面で，このときは65536色の中から16色を選んで使

うことができます。X68000のグラフィック能力を最大に生かせるのが512×512の画面で，65536色の同時表示ができます。

このほか512×512ドットの画面では，16色しか色を使わないモードと256色使うモードを選択できます。どうしてこんな色数が少ない画面があるのかというと，それは別のメリットがあるからです。これらのモードを指定すると，複数のグラフィック画面を同時に使えるようになるのです。複数の画面に別々に絵を描いておいて，それを交互に表示したり，続けて表示したり，あるいは重ね合わせて表示することが可能なわけで，大画面や多色数とは異なる世界を作り出すことができます。

●描画範囲とhome関数

次のようなグラフィック画面を考えてみましょう。

```
screen 0, 0, 1, 1
```

これは，表示画面を256×256に，実画面を1024×1024にする命令です。ここで，(0,0)～(1023,1023)に，つまり画面の対角線上にラインを引いてみることにしましょう。

```
line(0, 0, 1023, 1023, 15)
```

これでいいですね。line関数の最後のパラメータは色を表しています。16色しか使えない画面では，色番号の15は明るい白を意味します。画面を見ると確かに白い線が現れていますね。

さてここで質問です。画面に表示されないところもちゃんと線が引かれているのでしょうか？　X-BASIC はエラーを出しませんので，ちゃんと引かれているように思えますね。確かめてみましょう。

screen命令を実行した直後，画面左上隅の座標は(0,0)になっています。home 関数は，この画面左上隅の座標を変更する関数です。

```
home(0, 100, 100)
```

を実行してみてください。これで画面左上隅の座標は(100,100)になり，画面には(100,100)～(355,355)の範囲が表示されるようになります。つまり，1024×1024の大きなグラフィック画面を，256×256の窓から覗いているような感じですね。home関数はこの窓の左上の座標を決めるものだと思ってもらえればいいでしょう。最初のパラメータ0は，複数枚のグラフィック画面のうち，どのグラフィック画面を動かすのかを意味しています。1024×1024の画面はひとつしか取れませんので，0を指定しているわけです。

窓を移動してみておわかりのように，線は途中で切れています。X-BASICは，初期状態では見えるところにしか描画しないのです。描画する範囲はwindow関数を使って自由に変更することが可能です[2]。グラフィックを消去する命令はwipe( )ですので，まず，

```
wipe( )
```

とやって途中でとぎれてしまった線を消し，今度は，

```
window(0, 0, 1023, 1023)
```

を実行してから線を引いてみてください。

初期状態で描画範囲が見える部分に限定されているのは，グラフィックの表示を速くするためです。逆に描画範囲の中だけが表示対象範囲となることを利用して，

```
screen  1, 3, 1, 1
paint(1, 1, rgb(20, 0, 0))
window(200, 200, 400, 400)
wipe( )
```

というようなこともできます。window命令により描画範囲が(200,200)～(400,400)に限定されているため，グラフィック全部ではなく，この範囲に表示されているものだけが消去されるのです。この状態で(100,100)－(500,500)に線を引いてみましょう。どうですか？　ウィンドウの外には線が引かれませんね。これがクリッピングです。

スクロールの実際

画面をスクロールさせると，ディスプレイに上から（あるいは下や左右から）新しい画面が流れてきます。では，1024×1024の画面に日本地図を表示しておいて，九州から北海道へ窓を動かしながら覗いていく様子を想像してみてください。逆にこれは，窓を固定しておいて地図を動かしていると考えることもできますね。つまり，home関数を使えば，簡単にグラフィック画面をスクロールさせることができるわけです。

リスト2はこの感覚を味わってもらおうと用意したプログラムです。1024×1024の画面に同心円を表示し，それを256×256の窓でスクロールさせてみました。単に決まった方向にスクロールするだけではつまらないですから，マウスの動きに合わせてダイナミックにスクロールさせることにしましょう。

10～30行はプログラムで使用する変数の宣言です。mxとmyは例によってマウスの座標を得るのに使います。xとyはhome関数に渡す画面左上隅の座標を計算するために一時的に使用します。そしてiはループカウンタとして使っています。40行は画面の設定です。表示画面は256×256，実画面は1024×1024，ディスプレイは高解像度でグラフィックの表示はONです。50行はさっきやったばかりですね。描画範囲を設定しています。

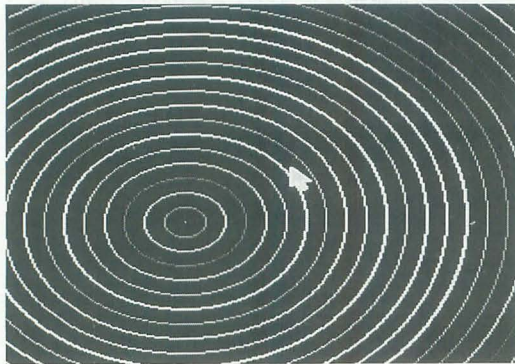
70～90行が同心円を描いている部分です。同心円は中心が同じで半径の異なる円を描けばいいので，ループカウンタiを使って，半径を10ずつ増やしな

2）window命令を使って描画する範囲を指定することを，ウィンドウを切るといいます。また，このようにして目的の範囲以外の場所に表示しないよう処理することを，クリッピング処理といいます。

がら円を表示します。さぁ，これで準備は終了。待望のスクロール部分です。

マウスを使う手順は前回やりました。mouse(0)はマウスを使うための儀式，mouse(1)でマウスカーソルが表示され，mouse(2)でマウスカーソルが画面から消えます。そして，130行はマウスカーソルが移動できる範囲を設定しています。

表示画面が256×256なのに，なぜわざわざマウスの移動範囲を(0, 0)～(255, 255)に設定しているのかと，疑問をお持ちでしょう。確かにマニュアルにも表示画面サイズによって移動範囲の最大値が決まるとあり，私もまさかこんなものが必要だとは思っていなかったのです。ところが実験してみると，なんと，画面の右端を越えてもマウスカーソルが動き続けるではありませんか。調べてみた結果，window関数を使ったあとでmouse(0)を実行すると誤動作するようです。mouse(0)をwindow関数の前に持ってくれば範囲設定は不要なのですが，逆にこれを利用して面白いことができるかもしれないと思い，このままにしておきます。

140行でマウスの位置を画面まんなかにし，150～200行がマウスの動きに合わせて画面をスクロールさせる部分です。170，180行に複雑な式がありますが，この2行が「回るんです」プログラムの命です。768という数字が謎かもしれませんね。これは，点(1023, 1023)を画面右下隅に表示するときに，home関数に与えればいい座標です。式の残りの部分は，mxが0のときxが0になり，mxが255のときxが768になるように調整しています。実際に値を入れて確かめてみてください。

リスト2　回るんです

```
 10 int mx, my
 20 int x, y
 30 int i
 40 screen 0,0,1,1               /* 256×256, 1024×1024
 50 window( 0, 0, 1023, 1023 )
 60 /*
 70 for i=0 to 50
 80   circle( 512, 512, i*10, i mod 15 + 1 )
 90 next
100 /*
110 mouse( 0 )                   /* マウス初期化
120 mouse( 1 )
130 msarea( 0, 0, 255, 255 )
140 setmspos( 128, 128 )         /* 座標設定にはmouse(1)が必要
150 while 1
160   mspos( mx, my )
170   x=(mx-256)*3+768           /* 感動の
180   y=(my-256)*3+768           /* スクロール
190   home( 0, x, y )
200 endwhile
```

遊び方はマウスを動かすだけです。ぜひ実行してみてください。もう，気分はマーダークラブDX。やはり地図などを眺めるのにいいのかもしれませんね。画面のまんなかあたりでマウスを円運動させると，ぐるぐる目が回る！　という，昔見た絵本のようなノスタルジーが，このプログラムの名前の由来です。こうしてみると1024×1024というのは使いでのある大きさです。大きな迷路を作りその中をさまようなど（絶対解けない気もするが）面白いかもしれませんね。

home関数を極める

screen命令のパラメータによっては，複数の画面を扱えることを説明しましたね。次のサンプルはこれを使ったゆらゆら芋虫，リスト3です。

このプログラムは4つのグラフィック画面をhome関数で動かすことによって，それぞれのグラフィック画面を凧糸でつないだかのように，ゆらゆら動かしてみようというプログラムです。いくつもの凧を一列につないで，蛇が大空を舞っているようなイメージを出している凧がありますね。ちょうどあんな感じです。もっともこの凧は4つしかつながっていないのですが。

複数のグラフィック画面を扱うため，X-BASICにはapage，vpageという関数が用意されています。apageはどのグラフィック画面に絵を描くか選択する関数，vpageは複数のグラフィック画面のうち，どれを実際に表示するか決める関数です。4つのグラフィック画面を扱う場合には，0～3の番号でそれぞれの画面を区別します。1024×1024の画面のようにひとつしか画面がない場合は0です。さっき使ったhome関数の最初のパラメータは，この画面番号だったのです。つまり，画面番号をapage関数のパラメータにすれば描く画面が指定できます。0番のページに描きたいのなら，

apage(0)

と指定します。また，vpage関数は，

vpage(&B○×△□)

として使うのが簡単でいいでしょう。○～□は順に3～0番の画面に対応していて，その画面を表示したいのなら1に，表示したくないのなら0にします。0番の画面と2番の画面を表示するのなら，

vpage(&B0101)

でOKです。4つの画面に少しずつ異なった絵を描いておき，vpage関数で1画面ずつ順に表示すれば，簡単なアニメーションが実現できますね。「&B」というのは２進数を意味するのですが，ここでは難しいことは抜きにして，こうすればいいのだとだけ覚えておいてください。

ではプログラムを見ながら説明していきましょう。最初は変数宣言と，画面，マウスの初期化です。60行でorgnという配列が宣言されていますね。これはグラフィック画面の左上隅の座標を，0～3番の各画面分用意しているところです。頭を動かせば，お尻が自動的についてくるようにするための変数です。具体的な使い方はあとで説明しましょう。

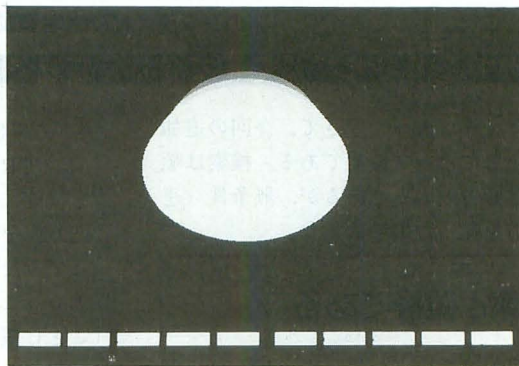
190～220行は色を設定しています。先月スプライトの色を設定したのと同様の方法で，1番の色は何，2番の色は何，……と決めています。そして240～280行で４つの画面それぞれに半径と座標を少しずつ変えて円を表示し，その中に色を塗ります。芋虫の本体はこの４つの円です。

320行で4つの円をすべて画面に表示します。さて，ここからが本番です。芋虫の頭は０番の画面に表示された円，お尻は３番の画面に表示された円です。頭が左に動けば，胴体とお尻はそれについて左に動きます。頭が右に動けばお尻も右です。これを実現するには，次のように考えれば簡単です。つまり，前回胴体が表示されていた場所にお尻を動かし，前回頭が表示されていた場所に胴体を動かせばいいのです。それぞれの画面に表示されている円はhome関数で動かしますから，前に胴体を表示していたときの画面左上隅の座標を，お尻の新しい左上隅の座標とすれば，お尻は胴体を追いかけて動くわけです。胴体も同様に，頭を追いかけて動くようにしてやります。

この処理をやっているのが340～400行です。orgn配列はそれぞれのグラフィック画面の左上隅の座標を記録しています。それを順次お尻の方から更新し，更新した座標にhome関数でスクロールさせれば，ほら，できあがりです。そして頭の新しい位置として，現在のマウスの座標を入れてやれば，芋虫はみごとマウスの動きに合わせてゆらゆら動いてくれることになります。

＊

プログラムは余計な飾りを省いて，エッセンスだけを取り出してあります。短いプログラムですからぜひとも入力してみてください。そして動かしたあとでもう一度プログラムを読んでみてください。自分で動作確認したあとでプログラムを見ると，ずっとよく理解できます。どうしてそうなるのかわからない部分があったり，どうやってプログラミングしているのかわからないところがあれば，納得がいくまで実行とプログラムの解読を繰り返してみてください。それが上達の一番の近道だと思います。

そして，プログラムが自分のものになったら，今度はプログラムに手を加える番です。頭からお尻まですべて円というのはつまらない，とお考えでしたら，スペースハリアーのドラゴンのような凝ったものを表示して動かしてみるのも面白いでしょう。改造は最良の教師なのですから。

リスト3　ゆらゆら芋虫

```
10 /*
20 /* ゆらゆら芋虫
30 /*
40 int i
50 int mx, my
60 int orgn( 3, 1 ) = {
70   +128, 128,
80   +128, 128,
90   +128, 128,
100  +128, 128
110 }
120 screen 1,1,1,1
130 mouse( 0 )
140 mouse( 1 )
150 msarea( 128,128, 383,383 )
160 setmspos( 256, 256 )
170 mouse( 2 )
180 /*
190 palet(1,rgb(24,24,0)) /* 黄
200 palet(2,rgb(0,24,24)) /* シアン
210 palet(3,rgb(24,0,24)) /* マゼンタ
220 palet(4,rgb(24,0,0))  /* 赤
230 /*
240 for i=0 to 3
250   apage(i)
260   circle(384,411-i*20,100-i*10,i+1) /* 384 = 256+128
270   paint(384,384,i+1)
280 next
290 /*
300 /* ゆらゆら開始
310 /*
320 vpage(&B1111)
330 while 1
340   i=3
350   while i>0
360     orgn(i,0) = orgn(i-1,0)
370     orgn(i,1) = orgn(i-1,1)
380     home( i, orgn(i,0), orgn(i,1) )
390     i = i-1
400   endwhile
410   mspos( mx, my )
420   orgn(0,0) = 384-mx
430   orgn(0,1) = 384-my
440   home( 0, orgn(0,0), orgn(0,1) )
450 endwhile
460 end
```

清く正しくズリズリと(その4)

Iwai Ippei
満開製作所 祝 一平

いよいよ，エディタ制作も終盤です。今月は文字列の検索・置換というエディタ必須の機能を付加します。それも，改行文字の検索というED.Xでもできないことをするスグレモノ(?)です。今回もズリズリした気配りと適度の手抜きの精神の一平氏，ぜひ叱咤激励のお便りを。

今月でやっと最後になるエディタであった。さて，今回の追加分は，文字列の検索と置換，そしてセーブ機能である。検索は順方向と逆方向の2種類，置換は順方向のみであるが，無条件（連続）置換と確認付き置換の両方を作ったのであった。

文字列の検索と置換である

リストであるが，E.Hには追加はない（ただしXwidth→96，Ywidth→32，MaxTextLine→1200などとするとエディタの編集範囲が広くなったりしてよろしいであろう）。EXTERN.Hにはリスト1を，VALUE.Cにはリスト2を追加していただきたい。さらに，先月号のA3.Cにリスト3のような追加をしてもらって，ファイル名を新たにA4.Cとしてもらう。そうして今月の中心はリスト4のSEARCH.Cである。コンパイルにはリスト5のようなバッチファイルがよろしいだろう。

増えたコマンドは，

ctrl-S	順方向検索
ctrl-R	逆方向検索
ESC+R	無条件置換
ESC + ctrl-R	確認付き置換
ctrl-X + ctrl-S	セーブ

である。検索/置換文字列の指定であるが，たとえば“DATA”という，文字列を検索したいのなら，[ctrl-S]とすると，画面の下のほうにプロンプトが出るので，

[D] [A] [T] [A] [ESC]

とタイプするのである。これは，たとえばED.Xなどのエディタであるならば，

[F4] [D] [A] [T] [A] [CR]

などとなっており，「検索文字列の終わり」は，[ESC]ではなく[CR]を押すようになっているのである。で，多分その結果であろうが，それらのED.Xでは改行を含む文字列を検索できないことになるのである。

んが，一度改行入りの文字列を検索できる便利さを知ると，それができないエディタはちょっと使う気にならないのである。これの便利なところは，たとえばCでは大抵の場合，関数は，

```
func1()
{
…
}
```

などと書くので，「その関数を呼び出しているところ」ではなく，「その関数を記述してあるところ」に一発で飛びたいときには，

[CR] [f] [u] [n] [c] [1] [(] [ESC]

を検索すればよいのである。どーだ便利だろう。そーゆーわけで，ED.Xを動かしてみた当初は「まさか」と思いつつ，あれこれ試してみたのだが，やっぱり改行は検索文字列に入れられないのだということを確認して，腹幌鰭晴(ハラホロヒレハレ)してしまった私であった。このことは置換についてもいえることである。

なお，文字を入力せずにいきなり[ESC]を押すと，前回検索/置換した文字列が採用される。すなわち，たとえばある文字列を何回か続けて検索するには，一度文字列を入力した直後では，

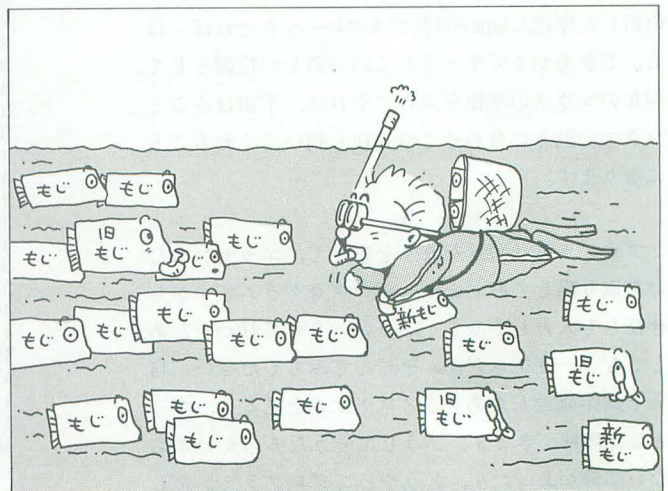
[ctrl-S] [ESC]

だけですむということになる。

それから確認付き置換は以下のようになっている。

[Y]	置換を実行して，次の置換対象文字列を検索する
[N]	置換は実行せず，次の置換対象文字列を検索する
[.](ピリオド)	中断し，置換開始時の位置に戻る
[ctrl-G]	中断する。置換開始時の位置には戻らない

無条件置換の実行中は，ctrl-Gで置換の中断ができるようになっている。それから，無条件置換なのであるから，本当は画面に表示しつつ置換する必要はないのであるが（処理が遅くなるので，むしろ表示しないほうがよい），節約の意味もあってほかのルーチンの使い回しをしたため，いちいち画面に表示している。勘弁していただきたい。

さて，検索の方法であるが，「検索文字列が複数の行にまたがっている可能性がある」わけであるから，アルゴリズムとしては，

1) まずは最初の1文字を捜す

2) その位置から，一致するかどうかをチェックする

ということをしている。次の行のデータを追加することによって，2行以上にまたがっている場合にも対応していることに注意。

今月のプログラムはこれといって特に説明すべき点はないのでここら辺でお開きにするのである。テキストの最後の付近での置換などではときどきバスエラーが起きるようである。少々のバグは勘弁願いたい。ではまた来月。

（編集部注：XC Ver.1.01を使用してください）

リスト1　EXTERN.H追加分

```
1: /* 第 3 回目 ↑ */
2:
3: extern char w_string0[];  /* 検索用文字列 兼 置換対象文字列 */
4: extern char w_string1[];  /* 置換用文字列 */
5:
6: /* 第 4 回目 ↑ */
```

リスト2　VALUE.C追加分

```
1: /* 第 3 回目 ↑ */
2:
3: char w_string0[MAXLINE];  /* 検索用文字列 兼 置換対象文字列 */
4: char w_string1[MAXLINE];  /* 置換用文字列 */
5:
6: /* 第 4 回目 ↑ */
```

リスト3　A3.C追加分（A4.Cになる）

```
******  m a i n ( ) の中のswitch文の中に追加 ******

++++ A3.C
          walk_flag = 1;   E_flag = 0;   break;
        }
++++ A4.C
          walk_flag = 1;   E_flag = 0;   break;
      case 'S':   /* search ### */
          do_S();
          walk_flag = 1;   E_flag = 0;   break;
      case 'R':   /* reverse search ### */
          do_R();
          walk_flag = 1;   E_flag = 0;   break;
        }
--------------------------------------------------------------

******  i n i t ( ) の中に追加 ******

++++ A3.C
    walk_flag = 1;       /* 連続してctrl-Kか */
}
++++ A4.C
    walk_flag = 1;       /* 連続してctrl-Kか */

    w_string0[0] = '¥0';   /* ### */
    w_string1[0] = '¥0';
}
--------------------------------------------------------------

******  d o _ X ( ) の先頭に追加 ******

++++ A3.C
{
    under_print("CTR-X:");   /* モードを示す */
++++ A4.C
{
    int i;        /* ### */
    char *p;    /* ### */
    FILE *fpw;    /* ### */
    char w[MAXLINE];

    under_print("CTR-X:");   /* モードを示す */
--------------------------------------------------------------

******  d o _ X ( ) の中のswitch文の中に追加 ******

++++ A3.C
    case 'S':   /* セーブ */
    /* まだだよーん */
        break;
++++ A4.C
    case 'S':   /* セーブ */
        fpw = fopen(filename0,"wt");    /* ### */
        if (fpw == NULL) {
            error("セーブできない");
            exit();
        }

        under_print("セーブ中…");
        i = fl;
        while(i != EOL) {
            p = jstrrchr(buff[i].data,Kaigyou);    /* 改行を捜す */
            if (p) {
                strcpy(w,buff[i].data);
                p = jstrrchr(w,Kaigyou);    /* 再度改行を捜す */
                *p++ = '¥n';
                *p = '¥0';   /* 改行＋nullをセットする */
                fprintf(fpw,"%s",w);
            } else {
                fprintf(fpw,"%s",buff[i].data);
            }
            i = next(i);
        }
        fclose(fpw);
        under_blanc();
        break;
--------------------------------------------------------------

        おっと、バグだぁ！

******  d o _ E S C ( ) の中のswitch文の中を訂正 ******

++++ A3.C
    under_blanc();       /* モード表示を消す */
    switch(com) {
    case '<':   /* テキストの先頭にジャンプ */
++++ A4.C
    under_blanc();       /* モード表示を消す */
    switch(toupper(com)) {
    case '<':   /* テキストの先頭にジャンプ */
--------------------------------------------------------------

******  d o _ E S C ( ) の中のswitch文の中に追加 ******

++++ A3.C
        break;
    default:
++++ A4.C
        break;
    case 'R':   /* 無条件置換 */
        replace();
        break;
    case 'R'-'@':   /* Y／N付置換 */
        replace_yn();
        break;
    default:
--------------------------------------------------------------
```

リスト4　SEARCH.C

```
 1: #include   <stdio.h>
 2: #include   <basic0.h>
 3: #include   <iocslib.h>
 4: #include   <conio.h>
 5: #include   <class.h>
 6: #include   "e.h"
 7: #include   "extern.h"
 8:
 9: /* 無条件置換 */
10: replace()
11: {
12:     int count = 0;
13:
14:     if (under_input2()) {/*置換元、置換先の文字列を入力する*/
15:         while(replace_sub(0) == 1) {
16:             count++;    /* 置換した個数を数えておく */
17:         }
18:         if (count) {/*置換が1個でもあったなら、画面書き換え*/
19:             cy = center_flush();
20:         }
21:         under_print("replaced ");
22:         printf("%d ",count);
23:     }
24:     under_close();    /* ウィンドウを切り直す */
25:     cursor();
26: }
27:
28: /* 確認しながら置換 */
29: replace_yn()
30: {
31:     int cx0,cy0,cxk0,cl0;
32:     int f,count = 0;
33:
34:     cx0 = cx;
35:     cy0 = cy;
36:     cxk0 = cxk;
37:     cl0 = cl;
38:     if (under_input2()) {/*置換元、置換先の文字列を入力する*/
39:         while((f = replace_sub(1)) == 1) {
40:             count++;
41:         }
42:         if (f == 3) {
43:           /* "."（ピリオド）が入力されたのでカーソルを戻す */
44:             cx = cx0;
45:             cxk = cxk0;
46:             cl = cl0;
47:             cy = center_flush();
48:         }
49:         if (!count) {
50:             under_print("対象が見付かりません");
51:         }
52:     }
53:     under_close();    /* ウィンドウを切り直す */
54:     cursor();
55: }
56:
57: /* 置換元、置換先の文字列を入力する */
58: under_input2()
59: {
```

```
60:   char ws[MAXLINE];
61:
62:     under_print("Replace [");
63:     ctr_prints(w_string0); /*コントロールコードを変換し表示*/
64:     printf("]<META> :");
65:     if (under_input(ws,w_string0) > 0) {
66:                     /* 有効な文字列を得た 0はダメ */
67:         strcpy(w_string0,ws);
68:
69:         under_print("widh [");
70:         ctr_prints(w_string1);
71:          /* コントロールコードを変換して表示する */
72:         printf("]<META> :");
73:         if (under_input(ws,w_string1) >= 0) {
74:             /* 有効な文字列を得た 0でも良い */
75:             strcpy(w_string1,ws);
76:             insert_cut_all(); /* 整行するして置換に備える */
77:             under_blanc();
78:             return(1);
79:         }
80:     }
81:     under_blanc();
82:     return(0);
83: }
84:
85:
86: /* 0=見付からなかった、1=見付かった：次の処理へ、
87:    2=中止：カーソルを戻さない、3=中止：カーソルを戻す */
88: /* 検索し、フラグに応じて確認し、置換する／しない */
89: /* 行った処理に応じてステータスを返す */
90: replace_sub(yn_flag)
91: int yn_flag;
92: {
93:     unsigned int c;
94:     char *p,*s,*s0;
95:     char wt[MAXLINE];
96:     UWORD l,l0;
97:     int len0,f,d;
98:     int cx0,cy0,cxk0,cl0;
99:
100:    UWORD r_l0,r_l1;
101:    int r_c0,r_c1;
102:
103:    int status = 0;
104:
105:    len0 = strlen(w_string0);   /* 長さをおぼえておく */
106:    if (iskanji(c = (w_string0[0] & 0xff))) {/* first char */
107:        c = (c << 8) | (w_string0[1] & 0xff);
108:    }
109:    p = &buff[cl].data[cct[cxk][0]]; /* 検索を開始する位置 */
110:    l = cl;                 /* 検索を開始する行 */
111:    cy0 = cy;               /* Y座標もおぼえておく */
112:
113:    /* 検索開始 */
114:    while(1) {
115:        if (kbhit() && (INKEY() == ('G'-'@'))) {
116:            return(2);   /* ^Gで中止 */
117:        }
118:
119:        if (s0 = s = jstrchr(p,c)) {   /* 掛かった */
120:            /* 本格的なチェックを行う */
121:            if (strlen(s) < len0) {   /* 長さが足りるか? */
122:                strcpy(wt,s);    /* 足りないなら足りる様にする */
123:                s = wt;
124:                l0 = l;
125:                while(strlen(s) < len0) {
126:                    l0 = next(l0);
127:                    if (l0 == EOL) {
128:                        *s = '¥0';    /* set dummy null */
129:                        break;
130:                    }
131:                    strcat(s,buff[l0].data);
132:                                    /* 後ろの行から持ってくる */
133:                }
134:            }
135:            if (!strncmp(w_string0,s,len0)) { /* 見つかった */
136:                /* 1行のs0(ポインタ)からlen0バイトの所 */
137:                r_l0 = cl = l;
138:                cx = cxk = 0;
139:                cursor0();   /* まずはその行の先頭へ */
140:
141:                d = (s0 - buff[cl].data);
142:                for(cxk = 0;d != cct[cxk][0];cxk++);
143:                cx0 = cct[cxk][1]; /* s0の差しているところへ */
144:                r_c0 = d;
145:                cl0 = cl;
146:                cxk0 = cxk;   /* 位置を保存 */
147:
148:                cy = cy0 + track(len0);   /* 置換の終端へ移動 */
149:
150:                if ((yn_flag) && (cy >= Ywidth-3)) {
151:                    /* 画面外へ出ているのなら、画面を書き直す */
152:                    under_blanc();
153:                    cy0 = center_flush();
154:                }
155:                r_c1 = track_c;
156:                r_l1 = track_l;   /* 削除すべき範囲が分かった */
157:
158:                cl = cl0;
159:                cx = cx0;
160:                cy = cy0;
161:                cxk = cxk0;
162:                cursor0();
163:
164:                if (yn_flag) {   /* 確認する */
165:                    under_blanc();
166:                    under_print("Replace '");
167:                    ctr_prints(w_string0);
168:                    printf("' with '");
169:                    ctr_prints(w_string1);
170:                    printf("'?");
171:                    cursor();
172:                    com = INKEY();   /* 1文字だけ入力 */
173:                } else {   /* 確認しないなら'Y'ということ */
174:                    com = 'Y';
175:                }
176:                switch(toupper(com)) {
177:                    case 'Y':   /* 置換 */
178:                    delete_region(r_l0,r_c0,r_l1,r_c1);
179:                    insert_str(w_string1);
180:                    status = 1;
181:                    break;
182:
183:                    case 'N':   /* 置換しない */
184:                    do_F();
185:                    status = 1;
186:                    break;
187:
188:                    case 'G'-'@':   /* 中止：カーソルは戻さない */
189:                    status = 2;
190:                    break;
191:
192:                    case '.':   /* 中止：カーソルを戻す */
193:                    status = 3;
194:                    break;
195:                }
196:
197:                break;
198:            } else {
199:        /* 先頭の1文字は一致したが、それ以降では駄目だった */
200:                if (iskanji((*s0++) & 0xff)) {
201:        /* 全角文字なら+2 */
202:                    s0++;
203:                }
204:                p = s0;   /* 進めたポインタからまた検索開始 */
205:                continue;   /* loop again */
206:            }
207:        }
208:        l = next(l);   /* 1行では見付からなかった */
209:        cy0++;
210:        if (l != EOL) {
211:            p = buff[l].data;   /* 次の行へ */
212:        } else {   /* 見つからなかった */
213:            break;
214:            status = 0;
215:        }
216:    }
217:    under_blanc();
218:    return(status);
219: }
220:
221: /* 逆方向検索 */
222: do_R()
223: {
224:    unsigned int c;
225:    char *p,*s,*s0;
226:    char ws[MAXLINE],wt[MAXLINE];
227:    UWORD l,l0;
228:    int len0,f,d;
229:    int cy0;
230:    char *pl;
231:    char *jstrrchr_x();
232:            /* 逆方向から文字を検索する：日本語対応版 */
233:
234:    under_print("Reverse search [");
235:    ctr_prints(w_string0);/*コントロールコードを変換して表示*/
236:    printf("]<META> :");
237:    if (under_input(ws,w_string0) > 0){/*有効な文字列を得た*/
238:        insert_cut_all();    /* 整行する */
239:        strcpy(w_string0,ws);
240:        len0 = strlen(w_string0);
241:        if (iskanji(c = (w_string0[0] & 0xff))){/*first char*/
242:            c = (c<<8) | (w_string0[1] & 0xff);
243:        }
244:        p = buff[cl].data;   /* 現在行の先頭 */
245:        pl = &buff[cl].data[cct[cxk][0]];   /* nullと見なす */
246:
247:        l = cl;
248:        cy0 = cy;
249:
250:    /* 検索開始 */
251:        while(1) {
252:            if (s0 = s = jstrrchr_x(p,pl,c)) {   /* 掛かった */
253:                /* 本格的なチェックを行う */
254:                if ((pl - s) < len0) {/*長さが足りるようにする*/
255:                    strcpy(wt,s);
256:                    s = wt;
257:                    l0 = l;
258:                    while(strlen(s) < len0) {
259:                        l0 = next(l0);
260:                        if (l0 == EOL) {
261:                            *s = '¥0';    /* set dummy null */
262:                            break;
263:                        }
264:                        strcat(s,buff[l0].data);
265:                    }
266:                }
267:
268:                if (!strncmp(w_string0,s,len0)) {/*見つかった*/
269:                    /* 1行のs0(ポインタ)に */
```

▶知っている人もいるかもしれませんが，Z'sSTAFF ver2.0にOMAKE.ZIMを発見しました。ASM_SOCEFILEのディレクトリの中です。attribの－Rオプションか，Z'sSTAFFのゴミ箱から拾ってください。まぁ，オタッキーね。左向きの顔とは㊙！

薄井　哲也（21）神奈川県

```
270:             cl = 1;
271:             cx = cxk = 0;
272:             cursor0();    /* まずはその行の先頭へ */
273:
274:             d = (s0 - buff[cl].data);
275:             for(cxk = 0;d != cct[cxk][0];cxk++);
276:             cx = cct[cxk][1];/* s0の差しているところへ */
277:             cy = cy0;    /* 新しいカーソル位置を求める */
278:
279:             if (cy < 0) {
280:               /* 画面外へ出ているのなら、画面を書き直す*/
281:                under_blanc();
282:                cy = center_flush();
283:             }
284:             cursor();
285:             break;
286:          } else {
287:             pl = s0;    /* 終わりを合わせて */
288:             continue;    /* loop again */
289:          }
290:       }
291:       l = before(l);    /* 前の行へ */
292:       cy0--;
293:       if (l != EOL) {
294:          p = buff[l].data;
295:          pl = NULL;
296:       } else {    /* 見つからなかった */
297:          break;
298:       }
299:    }
300: }
301:    under_blanc();
302: }
303:
304: /* 順方向検索 */
305: do_S()
306: {
307:    unsigned int c;
308:    char *p,*s,*s0;
309:    char ws[MAXLINE],wt[MAXLINE];
310:    UWORD l,l0;
311:    int len0,f,d;
312:    int cy0;
313:
314:    under_print("Search [");
315:    ctr_prints(w_string0);/*コントロールコードを変換して表示*/
316:    printf("]<META> :");
317:    if (under_input(ws,w_string0) > 0) {/*有効な文字列を得た*/
318:       insert_cut_all();       /* 整行する */
319:       strcpy(w_string0,ws);
320:       len0 = strlen(w_string0);
321:       if (iskanji(c = (w_string0[0] & 0xff))){/*first char*/
322:          c = (c<<8) | (w_string0[1] & 0xff);
323:       }
324:       p = &buff[cl].data[cct[cxk][0]];
325:       l = cl;
326:       cy0 = cy;
327:
328:    /* 検索開始 */
329:       while(1) {
330:          if (s0 = s = jstrchr(p,c)) {    /* 掛かった */
331:             /* 本格的なチェックを行う */
332:             if (strlen(s) < len0){/*長さが足りるようにする*/
333:                strcpy(wt,s);
334:                s = wt;
335:                l0 = l;
336:                while(strlen(s) < len0) {
337:                   l0 = next(l0);
338:                   if (l0 == EOL) {
339:                      *s = '¥0';    /* set dummy null */
340:                      break;
341:                   }
342:                   strcat(s,buff[l0].data);
343:                }
344:             }
345:
346:             if (!strncmp(w_string0,s,len0)) {/*見つかった */
347:                /* 1行のs0(ポインタ)からlen0バイトの所 */
348:                cl = l;
349:                cx = cxk = 0;
350:                cursor0();    /* まずはその行の先頭へ */
351:
352:                d = (s0 - buff[cl].data);
353:                for(cxk = 0;d != cct[cxk][0];cxk++);
354:                cx = cct[cxk][1]; /* s0の差しているところへ */
355:                cy = cy0 + track(len0);
356:                           /* 新しいカーソル位置を求める */
357:                cl = track_l;
358:
359:                if (cy >= Ywidth-3) {
360:                 /* 画面外へ出ているのなら、画面を書き直す */
361:                   under_blanc();
362:                   cy = center_flush();
363:                }
364:                cursor0();
365:                for(cxk = 0;track_c != cct[cxk][0];cxk++);
366:                cx = cct[cxk][1];
367:                cursor();
368:                break;
369:             } else {
370:             /* 文字列全体がマッチしないなら、1文字進める */
371:                if (iskanji((*s0++) & 0xff)) {
372:                   s0++;
373:                }
374:                p = s0;
375:                continue;    /* loop again */
376:             }
377:          }
378:          l = next(l);    /* 次の行へ */
379:          cy0++;
380:          if (l != EOL) {
381:             p = buff[l].data;
382:          } else {    /* 見つからなかった */
383:             break;
384:          }
385:       }
386:    }
387:    under_blanc();
388: }
389:
390: /* 逆方向から文字cを捜す */
391: /* ポインタp1がNULLの時は文字列の最後から */
392: /* ポインタp1がNULLでない時は、*/
393: /* p1の手前から（*p1を0x00と見なす） */
394: /* ポインタを返す */
395: char *jstrrchr_x(p,pl,c)
396: {
397:    char w[MAXLINE];
398:    char *wp;
399:
400:    if (pl) {    /* *plを'¥0'（文字列終了位置）と見なす */
401:       c = pl - p;    /* 長さ */
402:       strncpy(w,p,c);    /* 作業用の配列に転送 */
403:       w[c] = '¥0';    /* 終了記号をセット */
404:       if (wp = jstrrchr(w,c)) {
405:          wp = p + (wp - w);    /* ポインタを補正しておく */
406:       }
407:       return(wp);
408:    } else {
409:       return(jstrrchr(p,c));
410:    }
411: }
412:
413: /* 有効文字長を返す：無効なら－1を返す */
414: under_input(s,string_esc)
415: char *s,*string_esc;
416: {
417:    int c;
418:    char p[MAXLINE];
419:    unsigned int code;
420:
421:    s[c = 0] = '¥0';    /* 取り合えずデータを初期化 */
422:    while(1) {
423:       if (input(p,MAXLINE) == 1) {    /* 文字列入力 */
424:          com = *p;
425:
426:          if (com < ' ') {    /* コントロールコード */
427:             switch(com + '@') {
428:             case 'G':    /* 中止 */
429:                return(-1);    /* 無効 */
430:                break;
431:             case '[':    /* GO */
432:                if (c == 0) {
433:                /* いきなりESCなら、前の文字列を使う */
434:                   strcpy(s,string_esc);
435:                } else {
436:                   s[c] = '¥0';    /* 文字列終了をセット */
437:                }
438:                return(strlen(s));
439:                break;
440:             case 'H':    /* BS */
441:                if (c) {
442:                   c--;
443:                   if (jiszen(code = jlast(s) & 0xffff)) {
444:                      c--;
445:                      bs(2);    /* 全角削除 */
446:                   } else if (code < ' ') {    /* ctrl code */
447:                      bs(2);    /* ^?を削除 */
448:                   } else {
449:                      bs(1);    /* 半角削除 */
450:                   }
451:                   s[c] = '¥0';
452:                }
453:                break;
454:             case 'M':    /* CR */
455:                if (c < MAXLINE/2) {
456:                   s[c++] = '¥x81';
457:                   s[c++] = '¥xa5';
458:                   s[c] = '¥0';
459:                   printf("▼");
460:                } else {
461:                   beep();
462:                }
463:                break;
464:             default:    /* deafultとしてくさった！プンプン */
465:                   /* 他のコントロールコード */
466:                if (c < MAXLINE/2) {
467:                   s[c++] = com;
468:                   putchar('^');
469:                   /* コントロールコードを表示 */
470:                   putchar(com+'@');
471:                   s[c] = '¥0';
472:                } else {
473:                   beep();
474:                }
475:                break;
476:             }
477:          } else {
478:             if (c < MAXLINE/2) {
479:                s[c++] = com;    /* 普通の半角文字 */
```

▶X68000版のフラッピーの絵を見ていてどうしてフラッピーたちが上下にいけるのにストーンだけは下にいったきりなのか理不尽を感じていた人も積年の謎が解けましたね。まさか，斜面になっていたとは……。　　伊藤　孝真（20）愛知県

```
480:              s[c] = '¥0';
481:              putchar(com);
482:           } else {
483:              beep();
484:           }
485:           continue;
486:        }
487:     } else {    /* 文字列入力 */
488:        if ((c + strlen(p)) < MAXLINE/2) {
489:           strcat(s,p);    /* 連結 */
490:           c += strlen(p);
491:           printf(p);
492:        } else {
493:           beep();
494:        }
495:     }
496:   }
497: }
498:
499: /* カーソルの左側の文字をi個消す */
500: bs(i)
501: int i;
502: {
503:    for(;i>0;i--) {
504:       putchar(8);    /* カーソルを左へ */
505:       putchar(' ');     /* 消す */
506:       putchar(8);    /* もう一度カーソルを左へ */
507:    }
508: }
509:
510: /* コントロールコードを変換して表示する */
511: ctr_prints(s)
512: char *s;
513: {
514:    unsigned int c,p,code;
515:    char w[MAXLINE * 2];
516:
517:    c = p = 0;
518:    while(code = s[c++]) {
519:       if (code < ' ') {    /* ctrl code */
520:          w[p++] = '^';
521:          w[p++] = code+'@';
522:       } else {
523:          w[p++] = code;
524:       }
525:    }
526:    w[p] = '¥0';     /* end code */
527:    printf(w);
528: }
529:
530: /* 文字列の長さを返す */
531: /* 全角文字を1文字と数える */
532: /* 半角文字も1文字と数える */
533: jstrlen(s)
534: char *s;
535: {
536:    int l,c;
537:
538:    l = 0;
539:    while(c = *s++) {
540:       if ((iskanji(c)) && (!*s++)) {    /* 2バイト目が0 */
541:          return(l);
542:       }
543:       l++;
544:    }
545:    return(l);
546: }
547:
548: /* 文字列の最後の文字を返す */
549: /* 全角文字にも対応 */
550: jlast(s)
551: char *s;
552: {
553:    unsigned int c,c0,lc;
554:
555:    lc = 0;
556:    while(c = *s++) {
557:       if (iskanji(c)) {
558:          if (c0 = *s++) {
559:             lc = ((c << 8) & 0xff00) | ((c0) & 0xff);
560:          }
561:       } else {
562:          lc = c;
563:       }
564:    }
565:    return(lc);
566: }
567:
568: /* ウィンドウを切り直す */
569: under_close()
570: {
571:    B_CONSOL(0,0,Xwidth-1,Ywidth-4);
572:                         /* コンソールを標準状態にする */
573: }
```

リスト5 コンパイル用バッチファイル

```
1: cls
2: cc a4.c value.c alpha.c sub.c line.c search.c /W /Y /M /E
```

コマンド作成“基本”作法

Murata Toshiyuki 村田 敏幸

前回の最後で簡単に紹介しました「コマンドラインからのパラメータの取り込み」について詳しく解説します。コマンドラインから文字列を取り込む方法から，実際のプログラミングにおいての注意事項やエラーチェックの手法までを具体的に紹介しましょう。

前回は突っ走りすぎた（陳謝！）ので，今月は比較的軽め（かな？）の話題を取り上げよう。プログラムを“使いにくくしないためにはどうすればいいか”というノリで押してみようと思う（“使いやすくする”じゃないのが凄いでしょ）。プログラムにもいろいろあるわけだが，今回は起動時に指示を全部与え「ほら行け」と尻を叩くと勝手に処理をして帰ってくるようなプログラムを想定する。要するに，COMMAND.Xのプロンプトに続いてプログラム名と若干のパラメータを打ち込みリターンキーを押すとあとはユーザーに指示を仰ぐことなく処理を行って終了するようなプログラムだ[1]。

この種のプログラムではユーザーの指示はコマンドラインに打ち込んだ文字列という形でプログラムに伝えられる。プログラムはそれを解析することでユーザーが何をしたいのかを知る。コマンドライン文字列の解析処理ルーチンの作り方によっては使いにくいプログラムになってしまうこともありうるわけだ。そこで，このコマンドラインに与えられた文字列の扱い方を考えるのが今回の話の中心になる。

さて，たかがコマンドライン文字列解析とはいうものの，それなりにいろいろと考えさせる要素はある。マンマシンインタフェイスという言葉を持ち出せば，ことの重大さがわかろうというものだ。

昔は自分の作ったソフトを世に出そうと思ったら雑誌に投稿するか，ソフトハウスに売り込むか，自分でソフトハウスを作って売り出すか，それとも街頭でプログラムテープを配りまくるかぐらいしか手段がなかった。しかし，最近はパソコン通信などを通じて“どうぞお使いくださいソフト”として自由にばらまけるようになった[2]。せっかく作ったプログラムを自分ひとりでこそこそ使うのももったいないし，X68000の環境を整えるという意味でも，機会があればどんどん自作ソフトをばらまいてもらいたいと思う。その場合ユーザーの立場に立った細かな配慮といったものも必要になってくるだろう。ま，そういう希望もあって，今回の話は一応自作ソフトを他人に使ってもらうことを前提で進めたい。

使いにくいコマンド指定例

コマンドライン文字列の解析ルーチンの作成はそのプログラムを使う上での書式を設計することから始まる。ここでいう書式とはどんなパラメータをどんな形で指定するかといったことだ。ほかにどんなプログラムを使っているか，ユーザーのキータイプの癖，また好みの問題などからどのように書式を決めたら使いやすくなるかは一概に言えるものではない。しかし，客観的に見て明らかに使いにくい形というのはいくつか思いつくため，“使いにくいことをしない”ようにすればとりあえず“使いにくくないプログラム”にはなるだろう。

以下，よくない例を3つほど挙げてみたい。

1） OS標準コマンドの書式と極端に異なる

プログラムごとに使い方が全然違うようではユーザーに不要な負担をかけることになるから，使用法はある基準に従って統一されることが望ましい。何を基準とするかは意見が分かれるところかもしれないが，OSのコマンドプロセッサ（Human68kならCOMMAND.X）の内部コマンドや標準外部コマンドの使用法に準じるのがもっとも自然だろう。

たとえばHuman68kでは，複数のパラメータがある場合はスペースで区切って並べるようになっているのに“,”で区切ることにしたり，スイッチは“/A”のように指定するのが普通であるにもかかわらず“$A”などの見慣れない表記を用いることは避けるべきだ。

また，複数のパラメータがある場合の語順にも同じことがいえる。入力ファイルと出力ファイルの両方を指定する場合，COPYコマンドなどHuman68k上の大部分のプログラムでは，

　A>PROG　入力　出力

の形式を用いる。しかし，これを，

　A>PROG　出力　入力

のように通常と逆の順にすることはユーザーを混乱させるので避けたい。

1） あらかじめ指示を全部与えてプログラムを走らせることを“非対話的”とか“バッチジョブ的”という言葉で表現する。逆にプログラムを立ち上げてからユーザーの指示とプログラムの動作が交互に行えるようなものは“対話的”という。対話的というのも変な言葉だが，interactiveの訳らしい。

2） どうぞお使いくださいソフト：いわゆるPDSのことだ。PDSという，定義があいまい，もしくは誤用されている言葉を使いたくなかったためこういう表現をとった。

リスト1　PRINT.S

```
 1: *        コマンドラインパラメータの渡されかたを確認する
 2:
 3:          .include       doscall.mac
 4:          .include       const.h      *先月使った定数定義ファイル
 5: *
 6:          .text
 7:          .even
 8: *
 9: ent:
10:          lea.l    mysp,sp            *spの初期化
11:
12:          move.w   #'[',-(sp)         *[を表示
13:          DOS      _PUTCHAR           *
14:          addq.l   #2,sp              *
15:
16:          pea.l    1(a2)              *a2+1からの
17:          DOS      _PRINT             *  パラメータ文字列を
18:          addq.l   #4,sp              *  表示
19:
20:          pea.l    endmes             *]を表示して改行する
21:          DOS      _PRINT             *
22:          addq.l   #4,sp              *
23:
24:          DOS      _EXIT              *終了
25: *
26:          .data
27:          .even
28: *
29: endmes:  .dc.b    ']',CR,LF,0
30: *
31:          .stack
32:          .even
33: *
34: mystack:
35:          .ds.l    256                *スタック領域
36: mysp:
37:          .end
```

図1　リスト1実行例（↵はリターンキーを押した位置）

```
A>print test TEST test↵
[test TEST test]
A>print    test    TEST     test↵
[test    TEST    test]
A>print test TEST test          ↵
[test TEST test          ]
A>print test               >CON↵
[test               ]
A>print                    >CON↵
[]
A>print "<test>"↵
["<test>"]
```

3)　パラメータを“""”や“'”で囲んだ場合はこれらの記号そのものもコマンドライン文字列に含まれるので，プログラム側で取り除く必要がある。

リスト2　ARGO.S

```
 1: *        コマンドラインパラメータ解析
 2: *        パラメータを必要としない場合
 3:
 4:          .include       doscall.mac
 5:          .include       const.h
 6: *
 7:          .text
 8:          .even
 9: *
10: ent:
11:          lea.l    mysp,sp            *spの初期化
12:
13:          bsr      chkarg             *コマンドラインの解析
14:
15:          bsr      do                 *メイン処理
16:
17:          DOS      _EXIT              *正常終了
18:
19: *
20: *        メイン処理（今は何もしない）
21: *
22: do:
23:          rts
24:
25: *
26: *        コマンドラインの解析
27: *
28: chkarg:
29:          addq.l   #1,a2              *a2=コマンド行文字列先頭
30:          bsr      skipsp             *スペースをスキップする
31:          tst.b    (a2)               *余計なパラメータがあるか？
32:          bne      usage              *  そうなら使用法を表示
```

2)　キータイプ量が必要以上に多い

元々キーボードから文字列を打ち込むという行為自体が面倒がられるものだから，入力する文字列の長さはなるべく短くてすむようにしたい。スイッチにはたいてい英単語の頭1文字が使われるが，これをフルスペルで書くように強制するとか，ファイル名はフルパスで指定しなければならないとか，扱うファイルの拡張子は限定されているにもかかわらず省略が許されないといったプログラムは使っているうちにいらいらしてくるに決まっている。

3)　スイッチが覚えにくい

スイッチがたくさんあると，どのスイッチがどんな意味だったかわからなくなるものだ。通常，スイッチには意味のある英単語の頭文字が使われることが多いものだ。いうまでもないことだが，機能と関連のないでたらめな文字を使うのは避けたほうがよい。

また，趣味のプログラマは余計なスイッチを付けたがる傾向があるが，それぞれのスイッチが本当に必要かどうかは一度真剣に検討してみる必要があるだろう。

ちなみにそれぞれのスイッチはプログラムのほんの細かな部分の動作を制御するために用い，ひとつのスイッチを指定することによりプログラムの動作が大幅に変わるのは好ましくない。非常に極端な例を挙げると，スイッチを何も指定しない状態ではファイルをコピーするのに/Dスイッチを指定するとファイルを消去するようになるプログラムは機能別に2本のプログラムに分割したほうがよい。

さらに，どんなにわかりやすく書式を設定したとしても，ユーザーが暗記するのを期待するのではなく，簡単なヘルプ表示機能を用意することが望ましい。

こうやって言葉にしてみると当たり前のことばかりだが，以上の3点を“べからず集”の最上位にランクすることに読者も異存はないだろう。このべからず集に抵触しないことをプログラムを使いにくくしないための最低限の基準にしたいと思う。

パラメータ指定の注意点

書式を決めたらあとはそれに従って実際のパラメータ処理ルーチンを作るわけだ。ここでの注意点はとにかく“融通をきかす”ということに尽きる。

たとえば，パラメータ間の空白にしても，バッチの中から起動する場合などに備えてスペース文字（ASCIIコード20_H）だけではなくTABコード（09_H）も空白と見なすとか，空白はただひとつしか許さないのではなくいくつあっても構わないように作るとか，逆にファイル名とスイッチの間のように区切りが明らかな場合は空白の省略を認める，といった細工の余地がある。

スイッチに関してもHuman68kの標準形式である“/A”の形式だけではなく“-A”の形式（これは

Human68kの準標準形式といえる）も認め，また，大文字・小文字のどちらで指定しても受け付けるように作りたい。

さらに，スイッチとほかのパラメータとを並べる順序にもある程度の自由度が求められる。つまり，

A>PROG /A FILENAME

と指定しても，

A>PROG FILENAME /A

と指定しても受け付ける柔軟さがほしい。

あと，さっきも少し触れたように，そのプログラムで扱うファイルの拡張子が限定されている場合には拡張子の省略を認めるといった配慮も必要だし，もっと一般的なファイルを扱うプログラムの場合はワイルドカードも使えたほうがいいだろう。

プログラミングの実際

さて，プログラム例に入る前に確認の意味でコマンドライン文字列の構造を確認しておく。先月も話したように，Human68kではコマンドライン文字列はレジスタを介してプログラムに渡され，プログラム起動時のa2レジスタがこのコマンドライン文字列を指している。先頭の1バイトが文字列の長さを表し，その直後から00_Hが現れるまでが実際の文字列だ。このことからわかるように，コマンドライン文字列の最大長は255バイト（+00_Hの1バイト分）になる。

文字列はユーザーがコマンドラインに打ち込んだものから先頭のコマンド名（ファイル名）とそれに続くブランクを除いた部分がそのまま入る。パラメータ間の空白の数なども変わらず，末尾の余分なスペースすら残っている。なお，COMMAND.Xでリダイレクションやパイプラインを表すのに使われる文字，“<”，“>”，“|”以降はプログラムには渡されないことになっている。ただし，“"”か“'”で囲んでおくと，その間は無条件にプログラムに渡されることになっており，これを利用すればリダイレクト記号などもパラメータに含むことができる[3)]。

リスト1はこういったコマンドライン文字列の構造を確認するプログラムだ。与えられたコマンドライン文字列全体を“[”と“]”で囲んで表示する。図1の実行例を見てもらうと，リダイレクト記号以下が削られていることや末尾のスペースがそのまま残っていることが確認できるだろう。コマンド名とパラメータの間の空白が，その数に関わらず削られているということも発見できる。この構造を頭に入れたうえで，以下に示すようなパラメータの数に応じたコマンドラインパラメータ取り込み処理ルーチン例を見てほしい。

●パラメータを1個も必要としない場合（リスト2）

この場合はコマンドライン文字列を頭から無視しても構わないわけだが，パラメータが何もないことを確認したほうがいろいろな意味で安全だと思われる。もしなんらかのパラメータがある場合にはプロ

```
33:         rts
34:
35: *
36: *       コマンド行先頭のスペースをスキップする
37: *
38: skpsp0: addq.l  #1,a2               *ポインタを進め
39:                                     *繰り返す
40: skipsp:                             *サブルーチンはここから始まる
41:         cmpi.b  #SPACE,(a2)         *スペースか？
42:         beq     skpsp0              *  そうなら飛ばす
43:         cmpi.b  #TAB,(a2)           *TABか？
44:         beq     skpsp0              *  そうなら飛ばす
45:         rts
46:
47: *
48: *       使用法の表示&終了
49: *
50: usage:
51:         move.w  #STDERR,-(sp)       *標準エラー出力へ
52:         pea.l   usgmes              *  ヘルプメッセージを
53:         DOS     _FPUTS              *  出力する
54:         addq.l  #6,sp               *スタック補正
55:
56:         move.w  #1,-(sp)            *終了コード1を持って
57:         DOS     _EXIT2              *  エラー終了
58:
59: *
60: *       メッセージデータ
61: *
62:         .data
63:         .even
64: *
65: usgmes: .dc.b   '機  能：○○を××します',CR,LF
66:         .dc.b   '使用法：ARG0',CR,LF
67:         .dc.b   0
68: *
69:         .stack
70:         .even
71: *
72: mystack:
73:         .ds.l   256                 *スタック領域
74: mysp:
75:         .end
```

リスト3 ARG1.S

```
 1: *      コマンドラインパラメータ解析
 2: *      パラメータとしてファイル名を1個必要とする場合
 3:
 4:        .include        doscall.mac
 5:        .include        const.h
 6: *
 7:        .text
 8:        .even
 9: *
10: ent:
11:        lea.l   mysp,sp              *spの初期化
12:
13:        bsr     chkarg               *コマンドラインの解析
14:
15:        bsr     do                   *メイン処理
16:
17:        DOS     _EXIT                *正常終了
18:
19: *
20: *      メイン処理（今は与えられたパラメータを表示するだけ）
21: *
22: do:
23:        pea.l   arg                  *パラメータを表示する
24:        DOS     _PRINT               *
25:        addq.l  #4,sp                *
26:        bsr     crlf                 *改行する
27:        rts
28:
29: *
30: *      改行する
31: *
32: crlf:
33:        pea.l   crlfms
34:        DOS     _PRINT
35:        addq.l  #4,sp
36:        rts
37:
38: *
39: *      コマンドラインの解析
40: *
41: chkarg:
42:        addq.l  #1,a2                *a2=コマンド行文字列先頭
43:        bsr     skipsp               *スペースをスキップする
44:        tst.b   (a2)                 *パラメータがあるか？
45:        beq     usage                *  ないならパラメータが足らない
46:
47:        cmpi.b  #'/',(a2)            *パラメータの先頭が
48:        beq     usage                *  '/'か
49:        cmpi.b  #'-',(a2)            *  '-'であれば
50:        beq     usage                *  きっとヘルプが見たいのだろう
51:
52:        lea.l   arg,a0               *a0=パラメータ切り出し領域
```

```
 53:         bsr     getarg          *パラメータ1つをa0以降に取り出す
 54:
 55:         bsr     skipsp          *さらにスペースをスキップ
 56:         tst.b   (a2)            *パラメータがあるか?
 57:         bne     usage           *  あるならパラメータが多い
 58:
 59:         pea.l   nambuf          *DOSコールを使って
 60:         move.l  a0,-(sp)        *  ファイル名を
 61:         DOS     _NAMECK         *  展開してみる
 62:         addq.l  #8,sp           *
 63:         tst.l   d0              *d0が0でなければ
 64:         bne     usage           *  ファイル名の指定に誤りがある
 65:
 66:         rts
 67:
 68: *
 69: *       a2の指す位置からパラメータ1つ分を
 70: *       a0の指す領域へコピーする
 71: *
 72: getarg:
 73:         move.l  a0,-(sp)        *{レジスタ待避
 74: gtarg0: tst.b   (a2)            *1)文字列の終端コードか
 75:         beq     gtarg1          *
 76:         cmpi.b  #SPACE,(a2)     *2)スペースか
 77:         beq     gtarg1          *
 78:         cmpi.b  #TAB,(a2)       *3)タブか
 79:         beq     gtarg1          *
 80:         cmpi.b  #'-',(a2)       *4)ハイフンか
 81:         beq     gtarg1          *
 82:         cmpi.b  #'/',(a2)       *5)スラッシュ
 83:         beq     gtarg1          *
 84:         move.b  (a2)+,(a0)+     *  が現れるまで転送を
 85:         bra     gtarg0          *  繰り返す
 86: gtarg1: clr.b   (a0)            *文字列終端コードを書き込む
 87:         movea.l (sp)+,a0        *}レジスタ復帰
 88:         rts
 89:
 90: *
 91: *       コマンド行先頭のスペースをスキップする
 92: *
 93: skpsp0: addq.l  #1,a2           *ポインタを進め
 94:                                 *繰り返す
 95: skipsp:                         *サブルーチンはここから始まる
 96:         cmpi.b  #SPACE,(a2)     *スペースか?
 97:         beq     skpsp0          *  そうなら飛ばす
 98:         cmpi.b  #TAB,(a2)       *TABか?
 99:         beq     skpsp0          *  そうなら飛ばす
100:         rts
101:
102: *
103: *       使用法の表示&終了
104: *
105: usage:
106:         move.w  #STDERR,-(sp)   *標準エラー出力へ
107:         pea.l   usgmes          *  ヘルプメッセージを
108:         DOS     _FPUTS          *  出力する
109:         addq.l  #6,sp           *スタック補正
110:
111:         move.w  #1,-(sp)        *終了コード1を持って
112:         DOS     _EXIT2          *  エラー終了
113:
114: *
115: *       メッセージデータ
116: *
117:         .data
118:         .even
119: *
120: usgmes: .dc.b   '機　能：指定ファイルを××します',CR,LF
121:         .dc.b   '使用法：ARG1 ファイル名'
122: crlfms: .dc.b   CR,LF,0
123:
124: *
125: *       ワークエリア
126: *
127:         .bss
128:         .even
129: *
130: arg:    .ds.b   256             *パラメータ切り出し用バッファ
131: nambuf: .ds.b   91              *ファイル名展開用バッファ
132: *
133:         .stack
134:         .even
135: *
136: mystack:
137:         .ds.l   256             *スタック領域
138: mysp:
139:         .end
```

図2 nameckが返すファイル情報の形式

ドライブ名	2バイト	'A:','B:'のような形のドライブ名
パス名	65バイト	'¥'から始まり'¥'で終わる絶対パス+00_H
ファイル名	19バイト	18バイトまでのファイル名+00_H
拡張子	5バイト	'.'+3バイトまでの拡張子+00_H

グラムの使用法を表示し，パラメータを必要としないことをユーザーに教えよう。ヘルプ内ではこのプログラムが何をするプログラムでどういう書式で使うか（この場合はパラメータが何もなく単にコマンド名だけで起動できるということ）を示せば十分だ。

パラメータの有無がプログラムの実行に影響しない場合にもわざわざパラメータがあるかどうかをチェックしているのには2つの意味がある。ひとつは前にも述べたように安全のためであり，ユーザーの勘違いやミスタイプがもたらす（かもしれない）事故を防ぐという目的，もうひとつはほかのプログラムとの統一性という問題だ。たいていのプログラムでは意味のないスイッチ（よく使われるのは“/?”）が指定されると使用法を表示する慣例になっており，プログラムを初めて使うときや使い方を忘れたときには無意識のうちに，

A>PROG /?

とタイプする人も多いと思う。どんなプログラムでも“/?”が指定されたらヘルプを表示するよう統一されていれば，ユーザーは安心してプログラムを使うことができるというわけだ。ただ，現実にはHuman68kに用意されたプログラムでも使用方法を表示してくれないものもあるが[4]。

リスト2ではコマンドラインの解析はサブルーチンchkargで行っている。まずa2に1を足して，a2が文字列の先頭を指すようにする。それからサブルーチンskipspを呼び出してパラメータ文字列先頭の空白を読み飛ばす。リスト1で判明したように実際にはパラメータ文字列先頭の空白はあらかじめ削られているはずだが，これは明文化された規則というわけではないので万が一に備えてこういった処理を挟み，空白が絶対にないことを保証するわけだ。

そのあとでa2が指している（スペースでもTABでもない）文字が文字列の終端コード00_Hかどうかをtst命令で調べる。Zフラグが立てば文字列が何もないことになるから解析を終了し，メインルーチンに戻る。Zフラグが立たなかった場合は何らかの文字列があったことになるので使用法を表示するためにラベルusageに分岐する。usage以下では，使用法を表示してプログラムを終了する。ヘルプメッセージはラベルusgmes以下に用意している。

ところで，ヘルプメッセージは標準エラー出力に出すようにした。標準出力がリダイレクトされた場合にもヘルプメッセージが画面に表示されるようにしてあるわけだ。しかし，現実には“ヘルプメッセージをファイルに落としたい”といった要求もあるから，ヘルプメッセージを標準出力に出すか標準エラー出力に出すかは考えどころかもしれない。馬鹿らしいほど些細なことではあるが，大勢の人の意見を聞いてみたい気もする。

また，使用法を表示したあと終了コード1を持って終了し，処理が正しく行われなかったことを呼び出し側に知らせるようにしてある。ヘルプはエラーではないという考え方もあるだろうが，一般に正し

く処理が行われた場合のみ終了コード0で正常終了し，それ以外は1以上の終了コードを返したほうがユーザーのためになることが多いと思う。

●パラメータを1個必要とする場合（リスト3）

必ずただ1個のパラメータ（とりあえずファイル名ということにしよう）を伴って起動するプログラムの場合，チェック項目は3点ある。第1にパラメータがあるかどうか，第2にパラメータが正当かどうか，第3に余計なパラメータがないかどうかだ。いずれの場合もチェックに引っ掛かったら使用法か適切なエラーメッセージを表示して終了する。

パラメータがあるかどうか調べるところまではリスト2と変わらない。パラメータがあった場合は例によって“/?”かもしれないので，スイッチのチェックをする。このプログラムの場合はスイッチが何もないわけだから“/”か“-”が指定されたら即使用法の表示に飛んでよいだろう。

つぎに，パラメータひとつ分を適当なメモリ領域にコピーする。これはパラメータのあとに余計な空白やなんかが付いている可能性に備えてのことだ。なお，パラメータは最大255バイトだから，転送領域はそれにエンドコードの1バイトを足した256バイト以上を確保しておく必要がある。このパラメータの切り出し処理を行っているのがサブルーチンgetargで，このサブルーチンはa2が指すアドレスからスペースかTABか“/”か“-”か00_Hが現れる直前までをひとつのパラメータと見なして，a0でポイントしたメモリ領域（リストではラベルarg以下）へ転送する。扱いやすいように最後に文字列のエンドコード00_Hを書き込むのを忘れてはならない。

この時点でarg以下にはパラメータ1個分が切り出され，同時にa2はパラメータの直後を指すように更新されているので，a2の位置からさらにスペースを飛ばし，まだパラメータがあるかどうか調べ，もしあるようなら（パラメータの個数が多すぎるので）使用法の表示処理に分岐する。パラメータの個数が合っていることが確認できたら，今度はそのパラメータの正当性を調べる。どんな文字列を正当なパラメータと見なすかはプログラムによるわけだが，ここではファイル名である場合を取り上げる。

ある文字列がDOSの規則に従ったファイル名かどうかを調べるのは真面目にやろうとすると大変なので，ここはDOSに頼ろう。指定された文字列をとにかくファイル名だと見なして，DOSコールで読み込みモードでオープンしてみるのはてっとり早い方法だ。ファイルがオープンできなかったらDOSコールのエラー番号を調べれば，ファイル名に異常があったかどうかがわかる。

もう少しスマートにやりたければ，DOSコール$FF29のnamestsか$FF37のnameckを使う手がある。どちらのDOSコールもファイル名をドライブ名，パス名などに展開するもので，返ってくる情報の形が少し違う以外は同じような機能と考えてよい。展開の過程でファイル名の形式に合わない部分が見つかったらエラーを返すようになっているから，ファイル名の正当性を調べるのにも利用できる。ここではnameckのほうだけ解説しておこう。

nameckは次のようにして呼び出す。

```
move.l  展開バッファアドレス,-(sp)
move.l  ファイル名へのポインタ,-(sp)
DOS     _NAMECK
addq.l  #8,sp
```

展開バッファは91バイト以上確保しておく。呼び出し後，このメモリ領域に図2に示すような形式でファイル名が展開される。相対パスでファイル名を指定した場合も絶対パスに展開されることになっている。このDOSコールはワイルドカードにも対応しており，“＊”が指定されると適切な数の“?”に置き換

4) Human68k上のプログラムでヘルプのない代表例はCOMMAND.Xで，起動スイッチのヘルプも内部コマンドのヘルプもない。COMMAND.Xの起動スイッチなんか一度CONFIG.SYSで指定してしまえばあとはめったに使うものではないが，だからこそヘルプが必要なのだ。また，ヘルプがあっても意味をなしていないのがED.Xで，起動スイッチのヘルプがどーゆーわけか編集コマンドのヘルプの奥底に埋もれている。基本的なところで勘違いしているといわざるを得ない悪い見本である。

リスト4 NAMETEST.S

```
 1: *          nameckの動作を確認するプログラム
 2:
 3:            .include       doscall.mac
 4:            .include       const.h
 5: *
 6:            .text
 7:            .even
 8: *
 9: ent:
10:            lea.l   mysp,sp             *spの初期化
11:
12:            pea.l   namebuf             *与えられたパラメータを
13:            pea.l   1(a2)               *  ファイル名と見なし
14:            DOS     _NAMECK             *  nameckで展開する
15:            addq.l  #8,sp               *
16:
17:            lea.l   hexbuf,a0           *d0.lを16進8桁に変換し
18:            bsr     itoh                *  a0の指す領域へ格納しておく
19:
20:            lea.l   prttbl,a0           *a0=テーブルの先頭アドレス
21:
22:            moveq.l #4-1,d1             *以下を4回繰り返す
23:
24: loop:      move.l  (a0)+,-(sp)         *見出し部分を表示
25:            DOS     _PRINT              *
26:            addq.l  #4,sp               *
27:
28:            move.l  (a0)+,-(sp)         *対応する内容を表示
29:            DOS     _PRINT              *
30:            addq.l  #4,sp               *
31:
32:            pea.l   crlfms              *改行する
33:            DOS     _PRINT              *
34:            addq.l  #4,sp               *
35:
36:            dbra    d1,loop             *d1.wが-1になるまで繰り返す
37:
38:            DOS     _EXIT               *正常終了
39:
40: *
41: *          d0.lを16進8桁を表す文字列へ変換し(a0)以降に格納する
42: *                  レジスタはみんな保存する
43: *
44: itoh:
45:            movem.l d0-d2/a0,-(sp)      *｛レジスタ待避
46:
47:            moveq.l #8-1,d2             *以下を8回繰り返す
48:
49: itoh0:     rol.l   #4,d0               *d0.lを左に4ビット回転する
50:                                        *4ビットは16進1桁分！
51:                                        *たとえば
52:                                        *  $1234ABCD  →  $234ABCD1
53:            move.b  d0,d1               *d0の下位バイトをd1に取り出し
54:            andi.b  #$0f,d1             *下位4ビットを残してマスクする
55:                                        *d1にはd0.lを16進で表したときの
56:                                        *  最上位桁が入っている
57:            addi.b  #'0',d1             *ここで数値から16進を表す文字へ
58:            cmpi.b  #'9'+1,d1           *  変換する
59:            bcs     itoh1               *  0～9の場合は'0'を足すだけだが
60:            addq.b  #'A'-'0'-10,d1      *  A～Fの場合はさらに補正が必要
61:
62: itoh1:     move.b  d1,(a0)+            *変換した文字をしまう
63:
64:            dbra    d2,itoh0            *d2.wが-1になるまで繰り返す
65:
66:            clr.b   (a0)                *文字列終端コードを書き込む
67:
68:            movem.l (sp)+,d0-d2/a0      *｝レジスタ復帰
69:            rts
```

```
 70:
 71: *
 72: *           データ
 73: *
 74:         .data
 75:         .even
 76: *
 77: prttbl: .dc.l    mes1,hexbuf        *見出しとその内容の
 78:         .dc.l    mes2,drive         *対応を示したテーブル(表)
 79:         .dc.l    mes3,name          *見出しのアドレスと内容のアドレスが
 80:         .dc.l    mes4,ext           *1組になっている
 81: *
 82: mes1:   .dc.b    'NAMACKの戻り値(d0.l):',0
 83: mes2:   .dc.b    '        パス名:',0
 84: mes3:   .dc.b    '       ファイル名:',0
 85: mes4:   .dc.b    '        拡張子:',0
 86: crlfms: .dc.b    CR,LF,0
 87:
 88: *
 89: *           ワーク
 90: *
 91:         .bss
 92:         .even
 93: *
 94: namebuf:                            *nameckで
 95: drive:  .ds.b    2                  *  ファイル名が展開される
 96: path:   .ds.b    65                 *  領域
 97: name:   .ds.b    19                 *
 98: ext:    .ds.b    5                  *計91バイト
 99: *
100: hexbuf: .ds.b    8+1                *itohで16進文字列を格納する領域
101: *
102:        .stack
103:        .even
104: *
105: mystack:
106:        .ds.l    256                 *スタック領域
107: mysp:
108:        .end
```

5) 図2を見ると, Human68kのパス名の最大長が64文字までであることがわかる。パス名がこれより長くなるような深い階層ディレクトリは構築できないということだ。Human68kのマニュアルには階層ディレクトリの深さに制限はないような書き方がしてあるが, 現実にはこのような部分で目に見えない制限がある。これはサブディレクトリに短い名前を付けるか長い名前を付けるかによって, 階層の深さの限界が変わってくることを意味している。さっそく実験してみよう。

える[5)]。

指定したファイル名が正しかったかどうかはリターン時のd0.lを調べることでわかる。例によって, d0.lが負の数であればエラーでありファイル名が正しくなかったことを表し, d0.lがFF_H (=255) であればファイル名が指定されなかった (たとえば, ドライブ名のみ指定された) ことを表す。また, d0.lが0であればファイル名が正しく指定されたことを表し, d0.lが負でも, 0でも, FF_Hでもなかった場合はワイルドカードが指定されたことを表す。

よって, 単に ファイル名の正当性を調べたい場合でかつワイルドカードも認めないのであればd0.lが0かどうかだけを見ればよいことになる。

●nameckの動作を確認する(リスト4)

さて, nameckにはこれからもお世話になるので, より細かい動作を調べておこう。リスト4がテスト用プログラムになっている。与えられたコマンドラインパラメータをそのままnameckに渡し, 戻り値がいくつか(16進表記), どのように展開されたかを表示する。

リスト4では初登場の命令を2つ使っている。最初のdbraはループ制御用の命令で, なんでいまごろ出てきたのかというほど使用頻度が高い命令だ。すかさず頭に入れておこう。というわけで詳しくはコラム参照のこと。もうひとつはrolという命令なのだが, こいつに関しては別の機会に取り上げたほうがよさそうなので今回はリスト中の注釈で勘弁してもらいたい。

●パラメータを2個必要とし, オプションとして/A, /Bの2つのスイッチがある場合(リスト5)

だいたいよくある規模のプログラムといったところか。ちょっと複雑そうだが, 大筋はリスト4と変わらず2個のパラメータをdbraのループで処理しているぐらいの差なので, ここではスイッチの扱いだけに注目してもらおう。特によく見てもらいたいのはスイッチのチェックをいつするかという点だ。

A>ARG2 /A FILE1 FILE2

A>ARG2 FILE1 FILE2 /A /B

のようにどこにスイッチを置いても正しく処理ができるように細工してある。

A>ARG2 /A FILE1 /B FILE2

なんていうのまで可能だし, よーく見てもらうと,

A>ARG2 /AFILE1/BFILE2

のようなべた書きすら通すようになっているのがわかるだろう (最後のパターンは副作用のようなものだが)。

スイッチのチェックを行うのはサブルーチンnextargだ。このサブルーチンは先頭の空白を読み飛ばしたうえで, スイッチがあればその処理を行い, そうでなければそのまま戻るという動作をする。どちらにしろ, このサブルーチンから戻ったときにはa2レジスタは空白でもスイッチでもない文字 (またはエンドコード00_H) を指すことになる。

空白を飛ばしたあと, まず先頭1文字が"-"か"/"かどうか調べる。そうでなければスイッチではないパラメータ (か00_H) だからそのまま戻る。"-"か"/"のどちらかであればスイッチのはずだからポインタを進め, 次の1文字を取り出す。スイッチが大文字・小文字のどちらで指定されても困らないように, ここでサブルーチンtoupperを呼んで, この1文字を大文字に変換しておく。

それから順に/Aスイッチかな? /Bスイッチかな? と調べ, どちらでもなければ使用法の表示に飛ぶ。どちらかのスイッチであれば, それぞれに対応した1バイトのワークにFF_Hを書き込む。このワークはいわばスイッチがONかOFFかを表すフラグであり, 0ならOFF, それ以外ならONを意味するものとする。後のメイン処理から必要に応じて,

```
    tst.b    Aflg
    bne      /AスイッチがONの処理へ
```

リスト5 ARG2.S

```
  1: *        コマンドラインパラメータ解析
  2: *        パラメータとしてファイル名を2個必要とし
  3: *        /A,/B2つのスイッチを持つ場合
  4:
  5:        .include       doscall.mac
  6:        .include       const.h
  7: *
  8:        .text
  9:        .even
 10: *
 11: ent:
 12:        lea.l    mysp,sp            *spの初期化
 13:
 14:        bsr      chkarg             *コマンドラインの解析
 15:
 16:        bsr      do                 *メイン処理
 17:
 18:        DOS      _EXIT              *正常終了
 19:
 20: *
 21: *         メイン処理(今は何もしない)
 22: *
 23: do:
 24:        rts
```

OFFの処理～

というように参照されることになるだろう。

さて，実際にはフラグを立てる直前にすでにそのスイッチがONになっているかどうかを調べる処理が挟まっている。これは，

A>ARG2 /A /A FILE1 FILE2

のように同じオプションが2度指定されるのをはじくためだ。故意に同じスイッチを複数回指定することは考えられないので，ユーザーの操作ミスと見なして早めに教えてあげるわけだ。また，このサンプルでは/Aスイッチと/Bスイッチを同時に指定しても構わないものとしているが，２種類のスイッチの意味が矛盾するような場合は同様のチェックを競合するスイッチに対しても行う必要があるだろう。

スイッチを1個処理したら，再びnextargの先頭に飛ぶ。もちろんこれは複数のスイッチが連続して指定されても処理できるようにするためである。

今後の拡張の指針

ここまでのパターンが飲み込めれば，あとはオプションやスイッチの数が変わってもこれらの応用で片付けることができるだろう。あとは場合に応じた細かな配慮の問題になってくる。最後に１点だけ，簡単な工夫の例を示そう。

●拡張子の省略を許す（リスト６）

そのプログラムがある決まった拡張子のファイルを扱うのであれば省略することを認め，プログラム側で適当に補ってくれたほうがユーザーに優しい。ここで，本当に拡張子のないファイルを指定したい場合もあるので，

A>PROG FILE.

のように末尾に“.”を付けることで拡張子のないファイルを表し，

A>PROG FILE

のように“.”もない場合に限り内部で拡張子を補うことにする。これはごく普通に用いられている区別の仕方だと思う。

拡張子があるかどうかを調べ，なければ補うという処理自体は単純な文字列操作になる。ファイル名の先頭から順に“.”を探し，見つからなければ文字列の後ろに用意しておいた拡張子を付け加えればよい。また，拡張子の有無を調べるのには，さっき出てきたnameckを使うという手もある。さっきのnameckの動作テストプログラムNAMETEST.Xで次の２つのパターンを試してもらいたい。

a>NAMETEST TEST.

a>NAMETEST TEST

上のパターンのようにファイル名がピリオドで終わっている場合はnameckで展開されたあとの拡張子は“.”1文字からなる文字列になっている。また，下の例のように拡張子がなく，ピリオドもない場合は展開後の拡張子は空文字列である。つまり，nameckにかけてから，ファイル名展開バッファの

```
 25:
 26: *
 27: *          コマンドラインの解析
 28: *
 29: chkarg:
 30:         addq.l  #1,a2           *a2=コマンド行文字列先頭
 31:
 32:         lea.l   arg1,a0         *a0=パラメータ切り出し領域
 33:
 34:         moveq.l #2-1,d2         *以下を2回繰り返す
 35:
 36: ckarg0: bsr     nextarg         *スペースをスキップし
 37:                                 *  スイッチがあれば処理する
 38:         tst.b   (a2)            *パラメータがあるか?
 39:         beq     usage           *  ないならパラメータが足らない
 40:
 41:         bsr     getarg          *パラメータ1つをa0以降に取り出す
 42:
 43:         pea.l   nambuf          *DOSコールを使って
 44:         move.l  a0,-(sp)        *  ファイル名を
 45:         DOS     _NAMECK         *  展開してみる
 46:         addq.l  #8,sp           *
 47:         tst.l   d0              *d0が0でなければ
 48:         bne     usage           *  ファイル名の指定に誤りがある
 49:
 50:         lea.l   256(a0),a0      *a0 = a0+256
 51:
 52:         dbra    d2,ckarg0       *d2.wが-1になるまで繰り返す
 53:
 54:         bsr     getarg          *さらにスペースを飛ばす
 55:         tst.b   (a2)            *パラメータがあるか?
 56:         bne     usage           *  あるならパラメータが多い
 57:
 58:         rts
 59:
 60: *
 61: *          スペースを飛ばしつぎのパラメータ先頭までポインタを進める
 62: *          スイッチがあれば処理してしまう
 63: *
 64: nextarg:
 65:         bsr     skipsp          *スペースをスキップ
 66:
 67:         cmpi.b  #'/',(a2)       *パラメータの先頭が
 68:         beq     nxarg0          *  /,-であれば
 69:         cmpi.b  #'-',(a2)       *  スイッチ
 70:         beq     nxarg0          *
 71:
 72:         rts                     *スイッチはもうない
 73: *
 74: nxarg0: addq.l  #1,a2           *'/'や'-'の分ポインタを進める
 75:         move.b  (a2)+,d0        *1文字取り出す
 76:         bsr     toupper         *大文字に変換しておく
 77:         cmpi.b  #'A',d0         *Aスイッチ?
 78:         beq     asw             *  そうなら分岐
 79:         cmpi.b  #'B',d0         *Bスイッチ?
 80:         beq     bsw             *  そうなら分岐
 81:         bra     usage           *無効なスイッチが指定された
 82: *
 83: asw:    tst.b   Aflg            *Aスイッチの二重指定?
 84:         bne     usage           *  そうならエラー
 85:         move.b  #$ff,Aflg       *AスイッチON
 86:         bra     nextarg         *つぎのスイッチがあるかもしれない
 87: *
 88: bsw:    tst.b   Bflg            *Aスイッチの場合と
 89:         bne     usage           *  やっていることは同じ
 90:         move.b  #$ff,Bflg       *
 91:         bra     nextarg         *
 92:
 93: *
 94: *          英小文字→英大文字変換
 95: *
 96: toupper:
 97:         cmpi.b  #'a',d0         *英小文字か?
 98:         bcs     toupr0          *
 99:         cmpi.b  #'z'+1,d0       *
100:         bcc     toupr0          *
101:         subi.b  #$20,d0         *小文字なら大文字に変換
102: toupr0: rts
103:
104: *
105: *          a2の指す位置からパラメータ1つ分を
106: *          a0の指す領域へコピーする
107: *
108: getarg:
109:         move.l  a0,-(sp)        *{レジスタ待避
110: gtarg0: tst.b   (a2)            *1)文字列の終端コードか
111:         beq     gtarg1          *
112:         cmpi.b  #SPACE,(a2)     *2)スペースか
113:         beq     gtarg1          *
114:         cmpi.b  #TAB,(a2)       *3)タブか
115:         beq     gtarg1          *
116:         cmpi.b  #'-',(a2)       *4)ハイフンか
117:         beq     gtarg1          *
118:         cmpi.b  #'/',(a2)       *5)スラッシュ
119:         beq     gtarg1          *
120:         move.b  (a2)+,(a0)+     *  が現れるまで転送を
121:         bra     gtarg0          *  繰り返す
122: gtarg1: clr.b   (a0)            *文字列終端コードを書き込む
123:         movea.l (sp)+,a0        *}レジスタ復帰
```

```
124:          rts
125:
126: *
127: *          コマンド行先頭のスペースをスキップする
128: *
129: skpsp0: addq.l  #1,a2               *ポインタを進め
130:                                     *繰り返す
131: skipsp:                             *サブルーチンはここから始まる
132:          cmpi.b  #SPACE,(a2)        *スペースか?
133:          beq     skpsp0             *  そうなら飛ばす
134:          cmpi.b  #TAB,(a2)          *TABか?
135:          beq     skpsp0             *  そうなら飛ばす
136:          rts
137:
138: *
139: *          使用法の表示&終了
140: *
141: usage:
142:          move.w  #STDERR,-(sp)      *標準エラー出力へ
143:          pea.l   usgmes             *  ヘルプメッセージを
144:          DOS     _FPUTS             *  出力する
145:          addq.l  #6,sp              *スタック補正
146:
147:          move.w  #1,-(sp)           *終了コード1を持って
148:          DOS     _EXIT2             *  エラー終了
149:
150: *
151: *          データ
152: *
153:          .data
154:          .even
155: *
156: Aflg:    .dc.b   0         */Aスイッチon/offフラグ (=0...off,<>0...on)
157: Bflg:    .dc.b   0         */Bスイッチon/offフラグ (=0...off,<>0...on)
158: *
159: usgmes:  .dc.b   '機  能 : 入力ファイルを××して'
160:          .dc.b             '出力ファイルに書き出します',CR,LF
161:          .dc.b   '使用法 : ARG2 入力ファイル  出力ファイル',CR,LF
162:          .dc.b   '         /A     ○○を無視します',CR,LF
163:          .dc.b   '         /B     △△を□□と見なします'
164: crlfms:  .dc.b   CR,LF,0
165:
166: *
167: *          ワークエリア
168: *
169:          .bss
170:          .even
171: *
172: arg1:    .ds.b   256                *パラメータ切り出し用バッファ1
173: arg2:    .ds.b   256                *パラメータ切り出し用バッファ2
174: nambuf:  .ds.b   91                 *ファイル名展開用バッファ
175: *
176:          .stack
177:          .even
178: *
179: mystack:
180:          .ds.l   256                *スタック領域
181: mysp:
182:          .end
```

リスト6

```
 1:          .text
 2:          .even
 3: *
 4: *        拡張子が省略されていたら
 5: *        適当な拡張子を補う
 6: *
 7: chkext:
 8:          tst.b   nambuf+86          *拡張子はあるか
 9:          bne     chkex0             *  あるなら何もしない
10:
11:          lea.l   arg,a0             *用意してある拡張子を
12:          lea.l   dext,a1            *  連結する
13:          bsr     strcat             *
14:
15: chkex0: rts
16:
17: *
18: *        文字列を連結する
19: *        a0=被連結文字列,a1=連結文字列
20: *
21: strcat:
22:          tst.b   (a0)+              *(a0)は0か?
23:          bne     strcat             *そうでなければ繰り返す
24:          subq.l  #1,a0              *行きすぎたから1つ戻る
25: strcpy:
26:          move.b  (a1)+,(a0)+        *1文字転送
27:          bne     strcpy             *終了コードまで繰り返す
28:          rts
29: *
30:          .data
31:          .even
32: *
33: dext:    .dc.b   '.$$$',0           *補う拡張子
```

拡張子に当たる部分(バッファ先頭から86バイト目)を調べ，00_Hか“ ”かで拡張子が省略されたかどうかを知ることができる。

これを利用した例をリスト6に挙げておく。このサブルーチンchkextはリスト3のパラメータ解析処理の直後，メインルーチンの直前から呼び出すように作ってある。リスト3の113行あたりにでもこのサブルーチンを挿入し，リスト3の14行の空行を，

bsr　　chkext

のように変更してアセンブルしてもらいたい。

サブルーチンchkextが呼ばれた時点ではすでにnameckの結果がnambuf以下に格納されているので，nambuf+86の1バイトをtst命令で調べ，00_Hであったならファイル名の末尾に“.$$$”という拡張子を付け加えている。文字列の連結処理には以前作ったサブルーチンを流用した。

●

ワイルドカードまで手が回らなかったのは残念だが，今回はこれぐらいにしておく。また近いうちに続編をやることになるだろう。

さて，プログラムが使いにくくなるかならないかはほんのちょっとした部分の差でしかない。ならば，わずかな手間を惜しまないで使う側の立場に立ってプログラムをどんどん作ってばらまき，X68000のプログラム資産を増やして，みんなで幸せになろうじゃないか。

と，無理やり締めたところで来月へと続く。なお，次回は“大規模なプログラムを作るための小技”を予定している。

dbra命令

dbra命令はマシン語にしては珍しく用途のはっきりした高級言語指向の命令で，一定回数繰り返すループを構成するのに用いられる。dbraの頭のdはDecrementのdであり，braは“あの”braである。

dbra　　データレジスタ，分岐先

のようにして使い，指定したデータレジスタから1を引き（レジスタの内容を更新して），結果が−1であればdbraの直後の命令の実行に移り，そうでなければ指定した分岐先に制御を移す。サイズは常にワードであり，データレジスタの下位ワードのみが使用される。このためdbraひとつでは65536回までのループしか構成できない。

標準的な使い方はたとえば次のようになる。

```
        move.w  #1000-1,d0
loop:   繰り返す処理
            :
        dbra    d0,loop
```

この例ではd0.wをループカウンタとして使い，1000回同じ処理を繰り返す。dbraはデータレジスタが−1になるまで分岐を繰り返すので，ループカウンタに使うレジスタはループ回数より1小さい値で初期化しておかなければならない。

なお，上の例で

```
        move.w  #999,d0
```

ではなく，

```
        move.w  #1000−1,d0
```

という書き方をしているのは，1000回のループであることを強調しているのだと思ってもらいたい。

マシン語カクテル in Z80's Bar

第5回——善司ソフトの神髄——

シナリオ：西川善司
特別監修：浦川博之　金子俊一
イラスト：山田純二

今日も今日とて夜が来ました。Z80's Barでは今月もスクロールの講座が開かれようとしています。といっても今回はひと味違い(?)，質・量ともに西川氏の本領が発揮されそうです。ココアソーダに酔うかマシン語に酔うか，泥船の上で飲んでいるつもりでご聴講を。

♬カラーンカロン（ドアベルが鳴る）
西川善司(以下善)：あ，どーもどーも。皆さんいらっしゃいますねー。
マスター(以下M)：西川君いらっしゃい。
ようこ(以下Yo)：あら，長老もご一緒で。
善：先月見かけなかったでしょ，だから，家に様子を見にいったんですよ。そうしたら，祈りの甲斐なく……。
一同：甲斐なく？
善：生きてました。
長老（以下老）：こ，こら！
善：と，まぁそれで一緒に来たんです。ときにマスター，そこの外人は誰です？
M：彼女はね，カナダから日本に留学生として来ていてね，先週からうちでアルバイトしているんだ。私の娘の友人なんだよ。
メアリー（以下メ）：Hello！
善：ハ，ハロー，アイ，アイ，あい。
M：お猿さんの歌を歌っているのか君は。
老：ふふふ，わしに任せなさい。あ～っ，BASIC, COBOL, Oh!X, ACE HD!　TURBO Z!
メ：What does he say？
善：おぉ，通じましたね。さすがは長老。
老：はっはっはっはっ。恐れいったか。
Yo&M：この2人は……。
メ：ワタシ，日本ノパソコンタイヘンキョーミアルノコトヨ。
善：この人，漢字カナ交じりでしゃべりますよ。
老：そんなことより，ほれ，ここに来るとき話していた聞きたいこととはなんなのじゃ，いったい。あ，ようこちゃん，わしにコーヒースカッシュ（注1）をくれ。
善：あ，私にはマカダミアチョコレートナッツチップ入りソーダ（注2）を。いやぁ，いま，我が「ゼンジソフト」（注3）で第2回作品を制作中なんですけどね，画面のスクロールがわかんないですよ。
M：あれ，それって先月，光君がやってなかったっけ？
Yo：いやだぁ，西川君たら，あのとき一緒にいたじゃないのぉ。
善：はぁ，ずっと寝ていたような気もしますけど。まぁ，聞いてください。彼が作ったのはグラフィックをスクロールさせるやつでしょ。私はテキスト画面をスクロールさせたいんですよ。
メ：SCROLL？　マキモノデスカ？　ワタシ，テマキズシ，ダイスキデース。
老：ふむ，しかし，おぬし，そんなことも，わからんでよくソフトを作れるのう。まぁ，よい，とくと説明してつかわそう。

テキストイジール

源光(以下光)：おい，そこの2人，さっきから聞いていればメアリーさんを無視しおって。Mary, sorry for their rudeness.
メ：ワタシキニシテイナイアリマース。
光：まあ，平安時代の女性は皆カナで書いたものだが……。
善：おや，光か。いつ生えたんだ？
光：さっきからいるわい。だいたい来たときに「皆さんいらっしゃいますねー」なんて言っていたくせに。
老：まぁまぁ，ところで西川よ，テキストをスクロールさせること以前にどうやって文字を画面に出すかを知っておるのか？
善：うんや。(きっぱり)
メ：ワタシ知ッテマース。TEXT VIDEO RAMニモジFONT DATAヤモジCODEヲカキコメバイイノデース。
善：おい，外人，少しは漢字を使え！　読みにくい，もとい聞き取りにくいぞ。
光：いや，でも言っていることは正しい。
老：そのとおりじゃ。X68000のようなテキストVRAM(以下VRAM)の場合はフォント自体のデータ，つまり“A”なら，

00011000_B
00100100_B
01000010_B
01111110_B
01000010_B
01000010_B
01000010_B
00000000_B

のようなビットパターンを書き込まなければならない。
光：X1やMZを始め，多くのパソコンでは文字コードをVRAMに書けばその文字が画面に出る。
M：X68000では1文字を画面に表示するのに沢山のデータを書かなくてはならないんだねぇ。
老：そうじゃ。X1などだと1バイト，漢字では2バイトの文字コードを書くだけでいい。
Yo：文字コードってなーに？
光：コンピュータのプログラミングにおいて非常に文字を扱うことが多い。そこで，アルファベットや数字，漢字などに，1文字1文字に対応した数字を与えて扱いやすくしたのさ。
Yo：その数字を文字コードっていうのね。
メ：Yes. Have you ever heard about “ASCII CODE”？
善：ようこちゃんのことがあー好きー，だって。こいつレズっ気があんのかぁ？
光：バカ！　ASCIIコードを知っているかと言ったんだ。メアリーの言うとおり文字コードに代表的なものとしてASCIIコード，JISコードがある。
Yo：よくマニュアルの後ろに載っているやつ？　もしかして。
M：でも，どうしてX68000もX1のようなVRAMにしなかったんでしょうね。少ない

データ量で文字が出せるのに。

老：X68000のようなVRAMだとオリジナルの文字や自由な大きさの文字や，絵のようなものも画面に出せる。

光：また，文字の上に文字をかいたりすることも簡単にできますしね。まあ，16ビットパソコン以上になるとこっちのVRAMを採用したパソコンが結構ありますよ。

善：まあ，それはそうだとして，スクロールとなんの関係が？

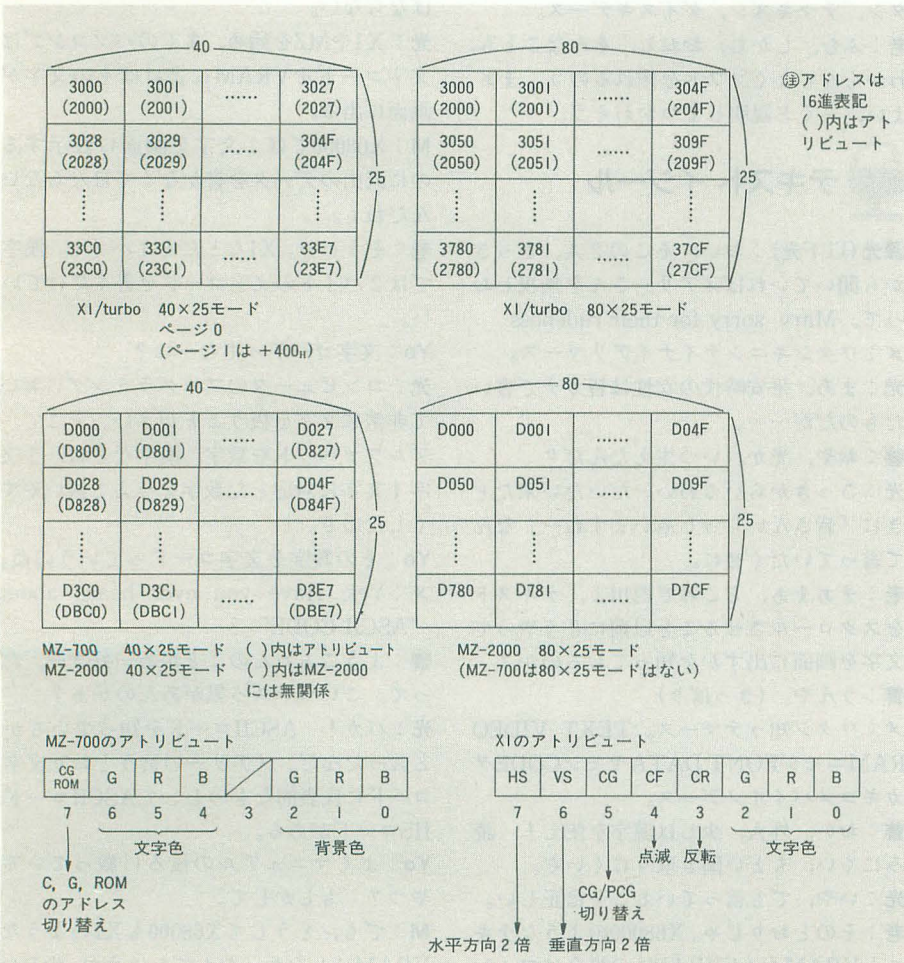
老：順を追って説明しているのじゃ。さて，具体的な文字の出力方法じゃがX1はI/Oポートの3000$_H$〜37FF$_H$がVRAMじゃ。2000$_H$〜27FF$_H$がアトリビュートじゃ。

メ：Oh!ワタシ，ヤキトーリ，スキアリマース。

善：カタカナで「うけ」を狙うなって！

M：アトリビュートって確か文字の色やなんかのことでしょ。

老：3000$_H$に"A"の文字コードである41$_H$を書いたとする。プログラムにすると，

```
LD    BC,3000H
LD    A,41H
OUT  (C),A
```

善：画面の一番左上に「A」がでるわけですね（図1参照）。

光：赤はカラーコードで2だから，2000$_H$に2をかく。

```
LD    BC,2000H
LD    A,2
OUT  (C),A
```

Yo：文字が赤くなるわけね。

老：これで分かったと思うが，VRAMのアドレス−1000$_H$がそのVRAMに対応したアトリビュートとなる。

メ：ワタシMZ-700知ッテアリマース。ワタシカナダデMZ-700ツカッテマーシタアリマース。MZ-700デハVRAMガD000$_H$カラアリマース。ATTRIBUTEハD800$_H$カラデース。

老：MZ-700のVRAMはバンク切り替えによってメインメモリのD000$_H$〜D7FF$_H$，アトリビュートはD800$_H$〜DFFF$_H$に割り当てられる。

善：すると，「A」をかく場合には，

```
LD   HL,D000H
LD   (HL),41H
```

ですね。でもバンク切り替えというのは？

図1　VRAMのアドレス

光：MZ-700はI/Oポート0E1$_H$，0E3$_H$への出力によって行う。

```
OUT  (0E1H),A
```

で，D000$_H$〜DFFF$_H$がVRAMやアトリビュートVRAMになる（注4）。

```
OUT  (0E3H),A
```

で，元のRAM，メインメモリに戻る。

M：Aにはなにが入っていればいいんですか。

メ：Oh!　ナンデモイーンデース。

善：モイーンってなんですか？

老：死ね。

光：そういえば同じような方式でVRAMにアクセスするマシンとして，MZ-2000なんかもありますね。

老：そういえば，そうじゃな。I/Oポート0E8$_H$を使って，バンク切り替えをするのは一緒だが，そのポートに何を出力してもいい，というのではない。第7ビットと，第6ビットが1だとテキストVRAMアクセス可能となる。バンク切り替えによって，MZ-700のようにD000$_H$〜D7FF$_H$がVRAMとなる。また，先ほどの2つのビッドが0だと元に戻る。

善：プログラムにすると，

```
LD   A,11000000B
OUT  (0E8H),A   ;バンク切り替え
LD   HL,0D000H
LD   (HL),41H
XOR  A          ;A=0
OUT  (0E8H),A
```

ですか。

老：いや，MZ-2000の場合，I/Oポート0E8$_H$のビットはほかの機能にも影響するから（注5），VRAMをメインメモリに持ってくるときは，

```
IN   A,(0E8H)
AND  00111111B ;下6ビット保存
OR   11000000B
OUT  (0E8H),A
```

戻すときは，

```
IN   A,(0E8H)
AND  00111111B ;下6ビット保存
OUT  (0E8H),A
```

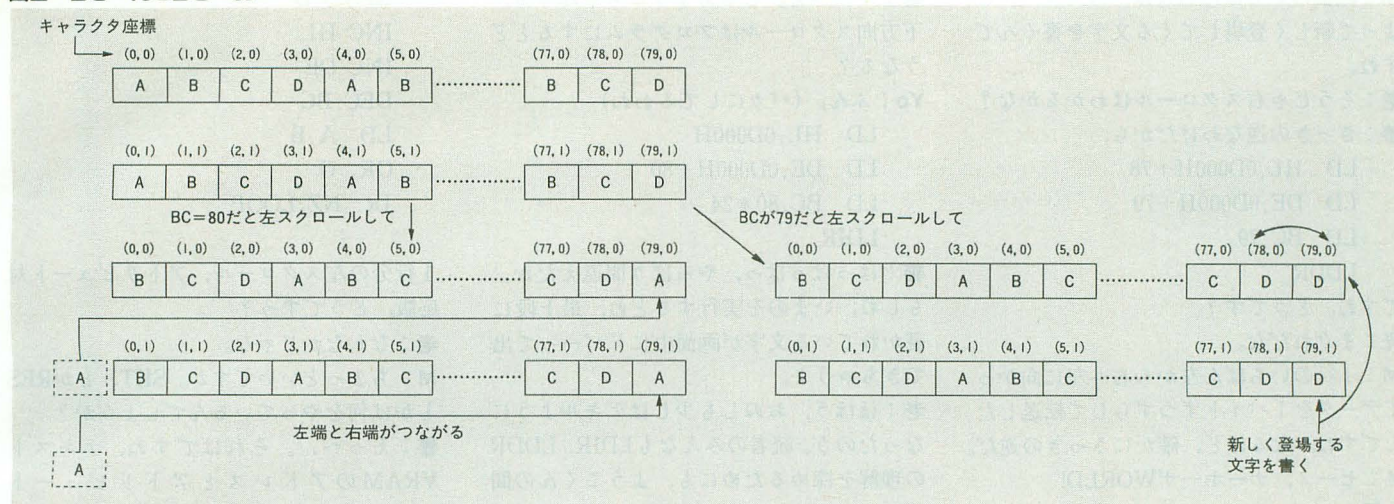
となる。

M：アトリビュートは？

光：MZ-2000では文字単位の色指定はできない。I/Oポート0F5$_H$で，グラフィックとのプライオリティ（優先順位）指定も兼ねて画面単位に文字色を決定する。たとえば，文字優先で文字色を緑にしたければ，

```
LD   A,4 ;カラーコード
OUT  (0F5H),A
```

グラフィック優先の文字色を緑にしたいな

図2 BC=79とBC=80

ら，先ほどのカラーコード4に8を加えた(第3ビットを立てた）12を出力すればいい。

```
LD   A,12
OUT  (0F5H),A
```

だね。

Yo：カラーコードってなーに？

メ：Oh!　エレクトロニクスノハッタツシタニホンニスムヒト，ソンナコトモワカラナーイノデアリマスカ。

Yo：発達ぐらい漢字でいいなさいよね。ふん。

善：カラーコードってBASICとかでよく使う0＝黒，1＝青，2＝赤，3＝紫，4＝緑，5＝水色，6＝黄色，7＝白ってやつでしょ。

老：そうじゃ。パソコンをかじったことが，ある人なら，みんな知っとるじゃろう。知らなかったら必ず覚えておいたほうがよいぞ。

メ：Oh!　ニホンジンパソコンカジルノデスカ？　I can't eat it.

Yo：すまきにして日本海に沈めてあげましょう。

メ：Oh! Wonderful!　スマキトノリマキトドッチガオイシイデスカァ？

ソンナワケデ横スクロールシマース

M：西川君じゃないけど，いい加減スクロールの話はどうなりました？

老：そうじゃな，基本的事項がわかれば説明は楽じゃからな。まずはMZのような$D000_H$から始まるVRAMでの左スクロールを考えよう。たとえば下のように，

```
D000  D001  D002  D003  …………
D1    D2    D3    D4    …………
```

VRAMにD1～Dnのデータが書かれていたとしよう。これをスクロールさせるにはどうしたらいいかな？

M：そうですね。$D001_H$から読んだデータD2を$D000_H$へ，$D002_H$から読んだデータD3を$D001_H$へ書くという動作を1行の終わりまで行えばよさそうですね。

善：そうしたら1行しかスクロールしないですよ。

メ：ソレハ簡単デース。1行オワッタラ次ノ行ノ初メカラ今ノ動作スレバヨロシイアリマース。

Yo：やっと漢字を使うようになったわね。

老：そのとおりじゃ。西川君，今，HLレジスタにVRAMの先頭アドレス$D000_H$が入っていたとするぞ。1行分のスクロールプログラムを組んでみなさい。

善：えーとぃ。

```
     LD   B,79
LOOP:
     LD   A,(HL)
     DEC  HL
     LD   (HL),A
     INC  HL
     INC  HL
     DJNZ      LOOP
```

どうですか？

光：まぁ，悪くないけどね。ブロック転送の命令を使えばもっと速く，しかもプログラムが短くなる。

M：ブロック転送？

メ：LDIR LDDRナドナドETC.

老：HLの示すアドレスから始まるデータBCバイト分DEの示すアドレスへ転送する，という命令だ。もし，LDIRをマシン語で動作を表すなら，

```
LOOP:
     LD   A,(HL)
     LD   (DE),A
     INC  HL
     INC  DE
     DEC  BC
     LD   A,B
     OR   C
     JR   NZ,LOOP
```

ってな感じかな。一応説明しておくと，

```
LD   A,B
OR   C
```

ってのはBとCレジスタが共に0であるか？　つまり，BCが0かどうかを試験している。そして，BCが0でないなら，まだループしていろってな感じだね。

M：LDDRは？

メ：INC HL ヲ DEC HL, INC DE ヲ DEC DE ニスレバOKデース。

M：メアリーあんた，Z80詳しいね。

メ：Of course，私ノ父親大工サンデース。母ハ新聞貴社デシタ。

善：意味不明だな。そいで，私のプログラムよりどう短くなって，速くなるっつうんですか？

老：こうじゃ。

```
LD   HL,0D001H
LD   DE,0D000H
LD   BC,79
LDIR
```

善：なるほどねぇ(2バイト長いケド……)。

Yo：あの，1行って80文字分あるんでしょ（注6）。なのにどうして，79なの？

光：それは，1行の最後と次の行の初めとつなげなくするためさ(図2)。つながってもいいんならば，80でもいいけどね。あ，ちなみにMZ-700やなんかだと，画面が40桁固定でしょう。そうなると，先ほどの老師のプログラムの「LD BC,79」は「LD BC,39」ってことになるね。(MZ-700系の人は以下も同様に変更して考えてね)

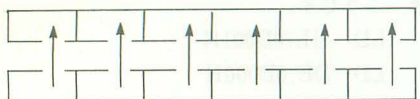
M：画面のいちばん右には，スクロールによって新しく登場してくる文字を書くんですね。
老：そうじゃ右スクロールはわかるかな？
善：さっきの逆なわけだから，

```
LD  HL,0D000H+78
LD  DE,0D000H+79
LD  BC,79
LDDR
```

ですね。どうです？
光：まぐれだな。
M：１行のいちばん左から右へ左に向かってデータを１バイトずつずらして転送したんですね。なるほど，確かにさっきの逆だ。
メ：セーノ，ナーホーザWORLD!
一同：……。
老：こ，これがわかると上スクロール下スクロールも同じことだ。

トイウワケデ上下スクロールシマース

Yo：どーしてどーして？
メ：OFFSETヲ，１カラ80ニスレバイイデース。
Yo：はぁ？
光：左右のスクロールでは隣のデータを自分の所に持ってくるっていう感じだったでしょ。それを，隣じゃなくて，上方向スクロールの場合は下から，下方向スクロールの場合は上から持ってくると考えればいいのさ。
メ：ツマリ，コウデアリマス。

（転送元→転送先トノ，アドレス差ハ１）
コレガ左スクロールノ場合ネ。
上方向スクロールのCASEハコウデース。

善：そうかそうか，じゃぁプログラムはこうなるわけだね。

```
LD  HL,0D000H+80
LD  DE,0D000H
LD  BC,80*24
LDIR
```

老：そうじゃ。80×24はわかるね。ようこちゃん。
Yo：１行が80文字分。縦方向は25文字分。だけど，最下段には新しく登場してくる文字などを書くから25−1=24なんでしょう？
メ：WELL DONE!
善：へへへぇ。じゃぁようこちゃんさぁ，下方向スクロールはプログラムにするとどうなる？
Yo：ふん，（バカにしてるわね）

```
LD  HL,0D000H
LD  DE,0D000H+80
LD  BC,80*24
LDIR
```

善：はっはっはっ，やっぱり間違えたか。もしね，いまのを実行するとね，最上段に書かれている文字が画面中にドバーって出てきちゃうよ。
老：ほほう，おぬしも少しはデキルようになったのう。読者のみんなもLDIR，LDDRの理解を深めるためにも，ようこくんの間違いプログラムを頭で実行してみなさい。下方向スクロールの正解はこうじゃ。

```
LD  HL,0D000H+80*23+79
LD  DE,0D000H+80*24+79
LD  BC,80*24
LDDR
```

これで，上下左右すべて終わったな。

アトリビュートモアルノコトネ

光：でも，このままでは不完全です。
善：は？　何がぁ？
光：色ですよ，色！　アトリビュートVRAMもスクロールさせないと色ずれが起きますよ。
老：そうじゃったな。しかし，LDIRなどで，テキストVRAM（文字など）をスクロールさせたあと，アトリビュートをスクロールさせたのではいくらマシン語とはいえ時間差が出てしまう。そうなると，今日の初めのほうで西川がやったような方法がいいといえるな。
M：と，いいますと？
メ：１バイトズツ，テキスト，アトリビュート，ト，ソレゾレヲテンソウスルデース。
善：ふふぅん。だいたいわかった。MZ-700の場合だと，テキストが$D000_H$，アトリビュートが$D800_H$からだから，

```
    LD  HL,0D001H
    LD  DE,0D000H
    LD  BC,79
LOOP:
    LD  A,(HL)
    LD  (DE),A
    SET 3,H
    SET 3,D
    LD  A,(HL)
    LD  (DE),A
    RES 3,H
    RES 3,D
    INC HL
    INC DE
    DEC BC
    LD  A,B
    OR  C
    JR  NZ,LOOP
      ⋮
```

１行分の左スクロール，アトリビュート対応版，どうですっ？
老：なかなかじゃな。
M：ちょっといいですか。SET，とかRESとかは何をやっているんでしょうか？
善：えっへ，。それはですね，テキストVRAMのアドレスとアトリビュートVRAMとは800_H離れてますよね。つまり，

$D800_H - D000_H = 800_H$
（テキスト）　（アトリビュート）

だから，テキストVRAMのデータを処理（5～6行目）上の演算をして，HL，DE共々そのテキストVRAMに対応したアトリビュートVRAMのアドレスにしなければならないですよね。
Yo：あぁ，それじゃ，「SET 3，なんとか」っていうのはその計算をしているのね。
M：でも，これが，DE=DE+800_HやHL=HL+800_Hだってのはよくわからないなぁ。
メ：Oh！ JAPANESE アタマニブーイ。「バイナリーレベル」デカンガエレバイイアリマス。
善：そうだね。まず，0800_Hを足すってことはさぁ，２バイトの演算になっちゃいそうだけど下位バイトのほうは変化しないでしょ，00だから。となると，上位バイトから８を足す計算をすればいいってことになる。
M&Yo：ふんふん。
善：で，テキストVRAM（の上位バイト）は$D0_H$から$D7_H$まで，アトリビュートは$D8_H$からDF_Hまで，これって２進表記すると

11010000_B～11010111_B（テキスト）
11011000_B～11011111_B（アトリビュート）

これらの第３ビットに注目してみて。
M：第３ビットってのは，右から４番目のやつだよね。いちばん右がビット０で，０，１，２，３ってね。これが？
Yo：あぁ，第３ビットがテキストでは０だけど，アトリビュートでは１だわ。
メ：イマゴーロキヅイタアリマスカァ。
善：これらのビットをいじってやれば演算命令を使ってやるより速いし，プログラムが短くなる。
老：そうじゃな，確かに考えてみれば上位４ビットのD_Hの部分は変化しないし，下位は，テキストでは０から７までしか変化し

ない。つまりテキストアドレスでは第3ビットは使っておらんということに着眼したわけじゃな。天晴れ！　天晴れ！
光：ははははははっ。さっきから聞いていれば長老まで。私ならこうします。

```
    LD  HL,0D001H
    LD  DE,0D000H
    LD  BC,79
    EXX
    LD  HL,0D801H
    LD  DE,0D800H
LOOP:
    LDI
    EXX
    LDI
    LD  A,B
    OR  C
    EXX
    JR  NZ,LOOP
      ⋮
```

このぐらい常識常識！　当たり前当たり前ぇ！
老：なるほど，裏レジスタを用いて$\pm 800_H$の演算を省き高速化を図ったのじゃな。少しわかりにくいプログラムになっておるが，さすが光君じゃな。
メ：ジャパニーズセコイコトトクイネ。

X1ノバアイハドウナルアリマスカ

メ：X1ノVRAMハI/Oネ。ジャパニーズノセコイワザミテミターイデース。
M：（こ，こいつは……）
Yo：I/Oっていうと，OUTとかINでやることになるのよね。
光：先月僕が使ったOUTIやINIなんていう命令のリピート版である，OTIRや，INIRなんて命令があるけど，あれらは使いものにならない。X1で使う場合にはね。
Yo：どーしてどーして？
メ：X1デハI/Oハ16Bit ADDRESS指定ネ。トナルト，BヲLOOP COUNTERトシテ使ッテイルコレラノ命令ハ，X1デ役ニ立ツコトハホトンドナイネ。
Yo：どうしてどうして？
メ：タトエバ，

```
    LD  BC,3000H
    LD  HL,0E000H
    OTIR
```

ヲ考エテゴランネー。
善：えーとぃ。最初にB＝B－1で$BC = 2000_H$になっちゃう。(HL)のメインメモリのデータ（この場合では$E000_H$の内容）がI/Oの2000_Hに出力される。2回目はB＝B－1で$BC = 1000_H$，3回目は$BC = 0000_H$？　VRAMなんか，通り越して，とんでもないアドレスになっちゃうな。
老：そうじゃな，そうなると，素直に組むしかなくなるな。

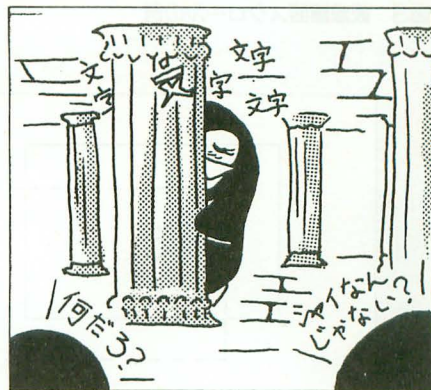
```
    LD  BC,3001H
    LD  D,79
LOOP:
    IN  A,(C)
    DEC BC      ；テキストスクロール
    OUT (C),A
    INC BC
    RES 4,B
    IN  A,(C)
    DEC BC      ；アトリビュートスクロール
    OUT (C),A
    SET 4,B
    INC BC
    INC BC
    DEC D
    JR  NZ,LOOP
      ⋮
```

光：ははははははっ。私なら裏レジスタを使って……。
善：もういいっつーの。
老：そうじゃな。上のやつの高速化は各自自由研究ということにして，さて，RESやSETが使われているがこれはもうみんなわかるな。
一同：はーい。BCの内容を，

$3000_H \longleftrightarrow 2000_H$

にしてまーす。
老：よろしい。BCに1000_Hを足したり引いたりしなくてはならないわけじゃが，下位バイトは00_Hで無変化。となると上位バイトにおいて，$\pm 10_H$を行うことになるが，第4ビットしか変化しないことに着眼する。
善：もういいですよ，そこまで言わなくても。ようするにさっきと同じことでしょ。
光：まあ，X1でゲームプログラムなんかを作る場合はPCGを使うことになるから，最初に全画面中PCG ONのアトリビュート（具体的には第5ビット＝1）とカラー7を書いたあとはいじらないことが多いから，
善：アトリビュートのスクロールはしなくてもいいと？
光：時と場合によるけどね。

ソシテ仮想画面

老：これでスクロールについては，ほぼいいじゃろう？
善：ええ，まあ。
メ：仮想画面ヲ使ウト8方向スクロールな

んかも簡単にデキルアルヨ。
善：こいつ，どこで日本語覚えたんだ？
M：仮想画面？？
善：あぁ，死んだ人を焼いて葬ることね。
Yo：それは火……。
M：相手にしちゃダメダメ。
老：仮想画面とは，メインメモリ上などVRAMとは別に画面に見立てたものをいう。
M：これがスクロールとなんの関係が？
老：うむ。あくまで一例じゃが，表示画面よりも大きい仮想画面を考える。たとえば$160 \times 100 = 3E80_H$を考える。マシン語プログラムとして，メモリ→VRAMの転送プログラムを用意して，表示開始位置をずらすだけで，あらゆる方向にスクロールすることができる（図3）。
メ：アトハ多重スクロールミタイナコトモ可能ネ。
善：それってどうやるの？
メ：仮想画面ヲ3枚用意スルネ。2枚ノ仮想画面ヲスクロールサセタアト，残ッタ1枚ニソレゾレ転送スルネ。ソコカラサラニVRAMへ転送スレバヨロシ（図4）。
老：おぉ，もうこんな時間じゃぞ。
M：閉店時間をとっくに過ぎてましたね。
光：それじゃ今日はこのへんで。
老：西川君今度くるときまでに何かを作ってくるようにな。君のためにみんなこんな遅くまで付き合ってくれたのじゃからな。
善：ふぇーい。

＊

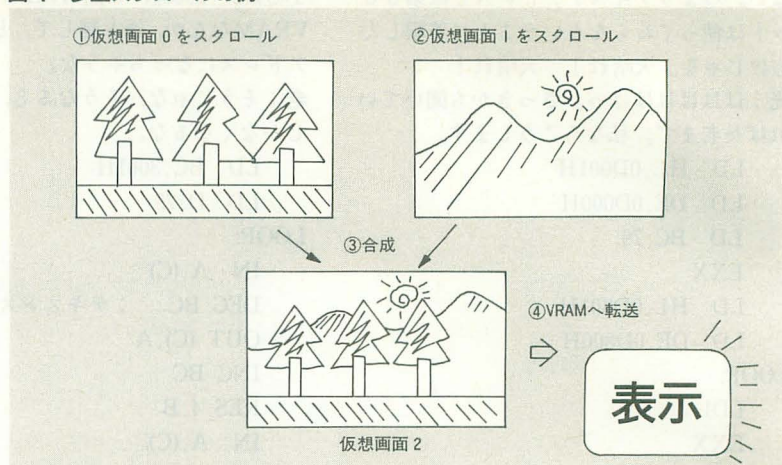
数日して……。

ワンダラーズフロムX1

善：なんとか形になったぞぉ。これで，いままでのツケを払えるなぁ。ああ，いま，日本の世の中は何時なのかしらん。おっと，読者の諸君，いいか，残された行数がないからプログラムの説明を一気にするぞ。あぁ，偉くなった気分だ。特にスクロールル

図3 仮想画面スクロールの例

図4 多重スクロールの例

ーチンの説明しかここでは書かないからね。あとはソースリストのコメントを読んでほしい。かなり細かくコメントを書いておいたし，大したことはしていないからわかると思うよ。

スクロールルーチンはリスト５のかなり後半に位置しているSCROLLというラベルのところ。まず，原理はメアリーが言っていた仮想画面を用いた方法を使っている。仮想画面を４枚使うことによって，３重スクロールを実現している。CALL ROLLとCALL ROLL2で仮想画面をスクロールさせ，CALL MIX，CALL MIX2で仮想画面を合成している。仮想画面0,1,2を合成してできた仮想画面３をテキストVRAMに転送して，スクロール処理終了。アトリビュートはプログラムの初めに設定したあとは変化しないので，特にスクロールルーチンではいじっていません。仮想画面→VRAMでは先月，光がやっていた「INC B，OUTI」で転送しています。コメントにもあるように，それぞれの仮想画面の左端と右端はつながるような処理をしています。

このままでもとりあえず見られる速度になっているけれど，もし，もっと高速化したかったり，もっと複雑なゲームにしたいならば，仮想画面のスクロール，転送にDMAを使うのが良いと思いますよ。何かわからないことがあったら，Oh!X編集部「マシン語カクテル質問コーナー」係にでも葉書をください。光や長老なんかが誌面の許す限り，答えると思うから。

最後にプログラムの実行方法。BASICはCZ-8FB01またはCZ-8CB01を起動してください。PCGデータを入力，適当なファイルネームでセーブしたあとRUN。その後MAPというBASICプログラムを入力して，同じく適当なファイルネームでセーブしてください。このMAPというプログラムは仮想画面データを作ります。できた仮想画面を（プログラム最後のSAVEMで）セーブしにいきますので，あらかじめカレントドライブにテープなりディスクなりをセットしておいてください。次に，“MMO.BIN”と“OBJ”いうマシン語ダンプリストを入力してください。MMO.BINのほうは単なるキャラクタのデータなのでどうしても面倒という方は入力されなくても結構です（一部表示が変になります）。

入力方法は，

CLEAR &HD000
MON ［RETURN］

“MMO.BIN”を入力するときは，

＊MF400 ［RETURN］

で入力を始めてください。

“OBJ”の場合は以下のようにします。

＊MD000 ［RETURN］

入力後チェックサムプログラムなどで間違いのないことを確認したら，

SAVEM“ファイルネーム”，スタートアドレス，エンドアドレス

リスト1 PCG

```
10 DEFCHR$(32)=STRING$(24,0)
30000 DEFCHR$(64)=HEXCHR$("001C3F632001000000000000C00000000011C7FE3E07F1900")
30010 DEFCHR$(65)=HEXCHR$("C020018181C00000000002000400002DF1662D99D9ECFB00")
30020 DEFCHR$(66)=HEXCHR$("1FDFEDBF8D2B00004C010000000180359FDFE5B98D237F00")
30030 DEFCHR$(67)=HEXCHR$("8CA274F97981000000804020408000548FA674F97BB7DF00")
30040 DEFCHR$(68)=HEXCHR$("8038FCC606980000000000030000000008038FEC707FE0C00")
30050 DEFCHR$(69)=HEXCHR$("000000000000000000441124161163000000000000000000")
30060 DEFCHR$(70)=HEXCHR$("000000000000000000220084200364054000000000000000")
30070 DEFCHR$(71)=HEXCHR$("000000000000000000009144AA7BFFFF0000000000000000")
30080 DEFCHR$(72)=HEXCHR$("000000000000000000001144AADEFFFF0000000000000000")
30090 DEFCHR$(73)=HEXCHR$("101028286C6CF6F6000008080C0C1616101028286C6CF6F6")
30100 DEFCHR$(79)=HEXCHR$("00000000000000000000AA00AA00CC003300000000000000")
30110 DEFCHR$(81)=HEXCHR$("212121212121212100880088008800886565656565656565")
30120 DEFCHR$(82)=HEXCHR$("0C0C0C0C0C0C0C0C00410041004100411C1C1C1C1C1C1C1C")
30130 DEFCHR$(83)=HEXCHR$("22222222222222220008000800080008A6A6A6A6A6A6A6A6")
30140 DEFCHR$(85)=HEXCHR$("00000000000000000036EEFFFFFFDD006000000000000000")
30150 DEFCHR$(86)=HEXCHR$("00000000000000000033F7FFFFFFBB630000000000000000")
30160 DEFCHR$(87)=HEXCHR$("000000000000000000004400110055AAB600000000000000")
30170 DEFCHR$(88)=HEXCHR$("000000000000000000004400120055AADA00000000000000")
30180 DEFCHR$(89)=HEXCHR$("F6F6F6F6F6F6F6F61616161616161616F6F6F6F6F6F6F6F6")
30190 DEFCHR$(94)=HEXCHR$("FF55008080D5FF5500AA0011000A00AAFF55008C9BD5FF55")
30200 DEFCHR$(95)=HEXCHR$("FF5500220253FF5500AA008800A800AAFF550032DA53FF55")
30210 DEFCHR$(96)=HEXCHR$("000304080030303000000001000200020003060C00393939")
30220 DEFCHR$(97)=HEXCHR$("000102040088888800200001000200020095 6AB400C8C8C8")
30230 DEFCHR$(98)=HEXCHR$("000E1F1F001F1F1F0040000000000000003E7F7F00DFDFDF")
30240 DEFCHR$(99)=HEXCHR$("00110804008383830084002100100010005329140 08B8B8B")
30250 DEFCHR$(100)=HEXCHR$("00804020001818180000000000400040080C06000383838")
30260 DEFCHR$(101)=HEXCHR$("00000000000000000602104B003401200000000000000000")
30270 DEFCHR$(102)=HEXCHR$("00000000000000000066000C200460080200000000000000")
30280 DEFCHR$(111)=HEXCHR$("000000000000000000DB00F700AA00FF0000000000000000")
30290 DEFCHR$(128)=HEXCHR$("04040400020100000100010000000000C0C0C0006030100")
30300 DEFCHR$(129)=HEXCHR$("4444440022118800020002000000201ACDCDCD0066339900")
30310 DEFCHR$(130)=HEXCHR$("0C0C0C000603810000000000800020A03C3C3C004E279300")
30320 DEFCHR$(131)=HEXCHR$("07070700020281008000800040002 0A87F7F7F003F3F9F00")
30330 DEFCHR$(132)=HEXCHR$("4A4A4A00D4D4A8000000000000010002AAEEEEEE00DCDCB900")
30340 DEFCHR$(133)=HEXCHR$("030303000C0C18008000800000000001284F4F4F009C9C3800")
30350 DEFCHR$(134)=HEXCHR$("00000000010204002200220044001190111111002386 8C00")
30360 DEFCHR$(135)=HEXCHR$("88888800102040002000200040000000989898003060C000")
30370 DEFCHR$(145)=HEXCHR$("04040404040404040091009100910091 4C4C4C4C4C4C4C4C")
30380 DEFCHR$(146)=HEXCHR$("40404040404040400008000800080008E5E5E5E5E5E5E5E5")
30390 DEFCHR$(147)=HEXCHR$("C1C1C1C1C1C1C1C10010001000100010CBCBCBCBCBCBCBCB")
30400 DEFCHR$(148)=HEXCHR$("50605060506050600002000200020002D1E1D1E1D1E1D1E1")
30410 DEFCHR$(149)=HEXCHR$("31313131313131310004000400040004737373737373737 3")
30420 DEFCHR$(150)=HEXCHR$("08080808080808080022002200220022999999999999999 9")
30430 DEFCHR$(160)=HEXCHR$("000001020004040400CC00F000A100E100010306000C0C0C")
30440 DEFCHR$(161)=HEXCHR$("00881122004444441A200080000200020099336600CDCDCD")
30450 DEFCHR$(162)=HEXCHR$("008103060000C0C0CA0200080000100010093274E003C3C3C")
30460 DEFCHR$(163)=HEXCHR$("008000400082828 2A8200080000000000952D6900DBDBDB")
30470 DEFCHR$(164)=HEXCHR$("00E8F4F4007A7A7AAA02000100000000 00F9FCFC00FEFEFE")
30480 DEFCHR$(165)=HEXCHR$("00180C0C000303032801000000800080000389C9C004F4F4F")
30490 DEFCHR$(166)=HEXCHR$("000402000000000090110044002200220 08C862200111111")
30500 DEFCHR$(167)=HEXCHR$("004020100088888 8001B004600220023 00C0603000989898")
```

でセーブしてください。

最後に“RUN”というBASICプログラムを入力して終わりです。このプログラムをRUNしてください。今まで入力したファイルを読み込むとゲームが始まります。

ゲームとはいっても大したものではなくただ，ジャンプで敵をよけるだけです（悲しぃーっ)。テンキーで左右移動，スペースでジャンプ。でも画面は一応多重スクロールのしかもオーバー処理付き（キャラクターが遠近によって見え隠れする)ですし(しつこいって?)，しかも▼§×？@の○★☆ですので，わかる人は笑ってください。あ？タイトルでわかるって？　失礼しました。

みなさんも何かゲームができたら送ってください。

ーつづくー

注1）実在するから怖い。
注2）強いて言うなら，ココアソーダ。西川君は350mlの一気飲みで寿命を縮めたといわれる。
注3）1989年9月号の「マシン語カクテル」参照。
注4）VRAMに切り替わる以外にもE000$_H$以降がキーやタイマ用のポートに切り替わる（メモリマップドI/O）。
注5）第5ビットは80/40桁モードの設定，0～4ビットはキーボード関係となっている。
注6）MZ-1200/700/1500などは画面がいわゆるWIDTH 40固定である。

リスト2 MAP

```
10 '@ABCDEFGHIJKLMNO
20 'PQRSTUVWXYZ[¥]^_
30 '`abcdefghijklmno
40 '▁▂▃▄▅▆▇█▕▐▌▍▎▉/
50 '─┼┬┤├┤┘└╭╮╰╯
60 ' ｡｢｣､･ｦｧｨｩｪｫｬｭｮｯ
70 CLEAR&HD000
80 MEM$(&HD000,16)=HEXCHR$("21 00 E0 01 00 30 11 00 05 ED A2 04 03 1B 7A B3")
90 MEM$(&HD010,3)=HEXCHR$("20 F7 C9")
100 CGEN1:CLS:CGEN1
110 LOCATE0,4:PRINTSTRING$(40,"GH")
120 LOCATE0,5:PRINTSTRING$(40,"WX")
130 LOCATE 0, 8:PRINT STRING$(40,"EF")
140 LOCATE 0, 9:PRINT STRING$(40,"UV")
150 LOCATE 0,10:PRINT STRING$(40,"ef")
160 POKE &HD002,&HE0:CALL &HD000
170  CLS
180 LOCATE 0,2:PRINTSTRING$(40,"^_")
190 LOCATE 0,3:PRINT "@ABCD     @ABCD     @ABCD     @ABCD     @ABCD     @ABCD
  @ABCD     @ABCD
200 FOR I=1 TO 9
210 LOCATE 0,3+I:PRINT " QRS       QRS       QRS       QRS       QRS       QRS
   QRS       QRS"
220 NEXT
230 LOCATE 0,13:PRINT "`abcd     `abcd     `abcd     `abcd     `abcd     `abcd
  `abcd     `abcd
240 LOCATE 0,14:PRINT "oooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
oooooooooooooooooooooooo"
250 POKE &HD002,&HE5:CALL &HD000
260  CLS
270 LOCATE 0,0:CGEN1
280 PRINT"▁▂▃▄▅▆▇█          ▁▂▃▄▅▆▇█          ▁▂▃▄▅▆▇█          ▁▂▃▄▅▆▇█
290 FORI=1TO 14
300 LOCATE0,I:PRINT" ┼┬┤├┤               ┼┬┤├┤               ┼┬┤├┤
 ┼┬┤├┤
310  NEXT
320 LOCATE 0,15:PRINT STRING$(4,CHR$(&HA0)+"｡｢｣､･ｦｧooooooooooooo")
330 POKE &HD002,&HEA:CALL &HD000
340 SAVEM "BG",&HE000,&HEEFF
350 CGEN
```

リスト3 RUN

```
10 WIDTH 80:INIT:PALET 4,0:CLS4:CSIZE2:PRW254:CLEAR&HD000
20 LOADM "BG",&HE000
30 LOADM "MMO.BIN",&HF400
40 LOADM "OBJ"
50 TIME=0
60 COLOR 3:LOCATE0,21:PRINT#0,"♥♥♥"
70 CALL &HD000
80 FOR I=2 TO 1 STEP -1
90 LOCATE I*2,21:PRINT"  "
100 CALL &HD01C
110 NEXT
111 LOCATE0,21:PRINT#0,"  "
120 COLOR 4:LOCATE 24,21:PRINT#0,"YOUR SCORE:";TIME*10
130 COLOR 5:LOCATE 24,23:PRINT#0,"TRY AGAIN ?[Y/N]"
140 A$=INKEY$
150 IF A$="N" THEN END
160 IF A$="Y" THEN CLS4:GOTO50 ELSE 140
```

リスト4 ダンプリスト

```
D000 CD 8A D2 CD A3 D2 21 26 : B2
D008 45 22 E2 D1 3E 02 32 E1 : 6D
D010 D1 3E 09 32 E4 D1 21 E7 : 07
D018 D1 22 E5 D1 AF 32 EF D1 : 4A
D020 CD B8 D2 CD 46 D3 FE 1B : 56
D028 CA C3 D3 FE 20 20 30 3A : 08
D030 E1 D1 B7 28 2A 3A E4 D1 : AA
D038 3D 32 E4 D1 28 02 3E 01 : 8D
D040 32 E1 D1 3A E4 D1 FE 05 : D6
D048 38 6A ED 4B E2 D1 CD F0 : 4A
D050 D1 2A E2 D1 01 50 00 B7 : B6
D058 ED 42 22 E2 D1 18 55 2A : 9B
D060 E2 D1 E5 3A E1 D1 FE 02 : 84
D068 20 1F 3A 95 D3 FE 34 20 : 33
D070 0B 7D D6 01 7C DE 45 38 : 36
D078 03 2B 18 0D FE 36 20 09 : B0
------------------------------------
SUM: A1 D9 B1 7A F2 F3 6A 1F BFC9

D080 7D D6 4C 7C DE 45 30 01 : 6F
D088 23 AF 32 E1 D1 3E 09 32 : 2F
D090 E4 D1 5D 54 01 50 00 09 : C0
D098 7D D6 4E 7C DE 45 38 07 : 7F
D0A0 6B 62 3E 02 32 E1 D1 22 : 13
D0A8 E2 D1 C1 7C B8 20 02 7D : 47
D0B0 B9 C4 F0 D1 3A E1 D1 FE : 28
D0B8 02 28 05 21 40 F6 18 17 : B5
D0C0 2A E5 D1 4E 23 46 23 7D : 37
D0C8 D6 EF 7C DE D1 38 03 21 : 4C
D0D0 E7 D1 22 E5 D1 69 60 ED : 46
D0D8 4B E2 D1 CD 16 D2 3A EF : DC
D0E0 D1 B7 CA 20 D0 C9 CD AA : 82
D0E8 D1 06 08 21 C9 D1 5E 23 : 1B
D0F0 56 7A B3 28 10 1B 7B D6 : 27
D0F8 6D 7A DE D4 30 03 11 00 : DD
------------------------------------
SUM: A0 83 C0 B8 A6 61 A4 14 0955

D100 00 72 2B 73 23 23 10 E6 : 4C
D108 11 D9 D1 21 C9 D1 3E 08 : BC
D110 08 7E 23 46 23 E5 6F 60 : C6
D118 01 50 00 B4 28 45 1A B7 : 43
D120 CA 58 D1 3D CA 48 D1 3D : 50
D128 CA 38 D1 36 49 CD 75 D1 : 65
D130 ED 42 36 20 AF C3 63 D1 : 2B
D138 ED 42 36 49 CD 75 D1 ED : AE
D140 42 36 20 3E 03 C3 63 D1 : D0
D148 ED 42 36 59 ED 42 36 49 : 6C
D150 CD 75 D1 3E 02 C3 63 D1 : 4A
D158 36 59 ED 42 36 49 CD 75 : 7F
D160 D1 3E 01 12 13 E1 08 3D : 5B
D168 C2 10 D1 CD 97 D3 7D FE : 55
D170 14 DC 8B D1 C9 C5 E5 01 : C0
D178 33 71 09 7C C6 80 47 4D : 03
------------------------------------
SUM: 94 0E A7 AD 27 75 CB BA 2237

D180 ED 78 28 03 32 EF D1 B7 : 39
D188 E1 C1 C9 21 C9 D1 11 D9 : 10
D190 D1 06 08 7E B7 28 06 23 : 65
D198 23 13 10 F7 C9 3E 49 32 : BF
D1A0 BC D4 36 BC 23 36 D4 AF : 5E
D1A8 12 C9 3E 20 11 CD D3 6B : 55
D1B0 62 23 01 4F 00 ED B0 12 : 84
D1B8 13 23 01 4F 00 ED B0 12 : 35
D1C0 13 23 01 4F 00 ED B0 12 : 35
D1C8 C9 00 00 00 00 00 00 00 : C9
D1D0 00 00 00 00 00 00 00 00 : 00
D1D8 00 00 00 00 00 00 00 00 : 00
D1E0 00 00 00 00 00 00 00 00 : 00
D1E8 F4 20 F5 40 F6 20 F5 00 : 54
D1F0 11 03 04 21 00 18 ED 69 : A7
D1F8 78 C6 08 47 25 C2 F6 D1 : 3B
------------------------------------
SUM: 5E 41 81 0A CA EA C0 6F E274

D200 CB F0 03 15 C2 F3 D1 16 : 6F
D208 04 69 60 01 4C 00 09 4D : 70
D210 44 1D C2 F3 D1 C9 78 D6 : FE
D218 10 59 57 D5 D9 C1 11 03 : 43
D220 04 D9 D9 ED 78 D9 FA 61 : 4F
D228 D2 CD 53 D2 CB F0 03 D9 : 5B
D230 03 15 D9 C2 22 D2 54 5D : 58
D238 60 69 01 4C 00 09 44 4D : B0
D240 6B 62 D9 16 04 60 69 01 : 8A
D248 4C 00 09 44 4D 1D D9 C2 : 9E
D250 22 D2 C9 16 18 04 ED A3 : 7F
D258 78 C6 08 47 15 C2 55 D2 : 8B
D260 C9 11 00 18 ED 59 23 78 : D3
D268 C6 08 47 15 C2 64 D2 C3 : E5
D270 2C D2 00 00 00 00 00 00 : FE
D278 00 00 00 00 00 00 00 00 : 00
------------------------------------
SUM: 68 D8 7C 8F 4A 21 71 93 C904

D280 00 00 00 00 00 00 00 00 : 00
```

```
D288 00 00 01 00 30 11 40 06 : 88
D290 3E 20 ED 79 CB A0 3E 27 : 94
D298 ED 79 CB E0 03 1B 7A B3 : 5C
D2A0 20 EE C9 06 F0 21 CD D3 : 8E
D2A8 36 20 23 10 FB 21 C9 D1 : 3F
D2B0 06 18 36 00 23 10 FB C9 : 4B
D2B8 11 00 E5 CD 16 D3 11 00 : BD
D2C0 EA CD 2A D3 11 00 EF 21 : D5
D2C8 00 E0 01 00 05 ED B0 11 : 94
D2D0 00 EF 21 00 E5 CD 04 D3 : 99
D2D8 CD E6 D0 11 C0 F2 21 CD : 34
D2E0 D3 01 F0 00 CD 07 D3 11 : 7C
D2E8 00 EF 21 00 EA CD 04 D3 : 9E
D2F0 01 40 31 21 00 EF 11 00 : 93
D2F8 05 04 ED A3 03 1B 7A B3 : E4
---------------------------------
SUM: 28 75 0B E4 97 7B C0 B6 3FC8

D300 C2 F9 D2 C9 01 00 05 7E : DA
D308 FE 20 28 01 12 23 13 0B : 9A
D310 78 B1 C2 07 D3 C9 3E 10 : DC
D318 08 1A 6B 62 23 01 4F 00 : 62
D320 ED B0 12 13 08 3D C2 18 : E1
D328 D3 C9 3E 10 08 13 1A F5 : 14
D330 1B 1A 6B 62 23 23 01 4E : 97
D338 00 ED B0 12 F1 13 12 13 : D8
D340 08 3D C2 2C D3 C9 C5 D5 : 69
D348 E5 3A 95 D3 32 96 D3 FB : 1D
D350 21 94 D3 16 E6 CD 6E D3 : 92
D358 CD 80 D3 F3 CD 77 D3 72 : 9C
D360 23 CD 77 D3 72 23 FB 3A : 04
D368 95 D3 E1 D1 C1 C9 CD 80 : F1
D370 D3 01 00 19 ED 51 C9 CD : C1
D378 8A D3 01 00 19 ED 50 C9 : 7D
---------------------------------
SUM: 0B 63 E8 8F 1E 40 4E 6C 4537

D380 01 01 1A ED 78 E6 40 20 : C7
D388 FA C9 01 01 1A ED 78 E6 : 2A
D390 20 20 FA C9 00 00 00 ED : F0
D398 5B AB D3 ED 4B AE D3 CD : 5F
D3A0 B0 D3 22 AB D3 ED 5F 32 : A1
D3A8 AD D3 C9 23 E1 00 83 03 : D3
D3B0 21 00 00 3E 10 29 CB 23 : 86
D3B8 CB 12 D2 BE D3 09 3D C2 : 48
D3C0 B5 D3 C9 CD 46 D3 FE 1B : 50
D3C8 28 F9 C3 23 D0 00       : D7
---------------------------------
SUM: 9C 19 31 5E 8A 73 73 F5 4806


F400 00 00 00 07 FF 7F 3F 1F : E3
F408 00 00 00 07 FF 7F 3F 1F : E3
F410 00 00 07 F8 00 80 40 20 : DF
F418 00 06 21 FE F9 FF FE FE : 19
F420 00 07 3E FF FF FF FF FF : 40
F428 07 39 DE 01 06 00 01 01 : 27
F430 00 10 F0 CF FF E4 A4 A5 : FB
F438 00 F0 3E 7F FF F6 E6 E7 : 6F
```

```
F440 F0 EE 1F 70 00 3F FF FF : AA
F448 00 00 00 80 00 00 00 00 : 80
F450 00 00 00 80 00 00 00 00 : 80
F458 00 00 80 40 80 00 00 80 : C0
F460 07 00 00 00 00 01 01 00 : 09
F468 07 00 00 00 00 01 01 00 : 09
F470 18 07 00 00 01 03 03 01 : 27
F478 FF FF 03 1F FF FF 9F 9F : 5C
---------------------------------
SUM: 1C 3A 14 21 7A 99 E9 07 CA6B

F480 FF FF 00 07 C7 C1 80 80 : 8D
F488 01 00 BC 27 C7 C1 A0 A0 : AC
F490 AD AC E0 C0 FE FF FC FC : EE
F498 FF FE 60 00 86 FF 78 00 : 5A
F4A0 FF FF 7E 3E 87 FF 7B 03 : BE
F4A8 00 00 00 20 50 00 00 00 : 70
F4B0 00 00 00 70 70 00 00 00 : E0
F4B8 80 00 70 F8 F8 F0 00 C0 : 90
F4C0 00 00 00 00 00 00 00 00 : 00
F4C8 00 00 01 03 00 01 07 03 : 0F
F4D0 00 00 02 04 01 07 0F 07 : 24
F4D8 00 00 00 04 28 00 00 00 : 2C
F4E0 1F 7F FF FE 3C C8 E0 FC : 7B
F4E8 E0 80 00 0F FE FC FC FE : 63
F4F0 00 00 02 00 00 00 00 00 : 02
F4F8 FF FC F3 01 0C 3F 1C 07 : 5D
---------------------------------
SUM: 29 A3 E1 CD C0 7A 1D EA 8DDB

F500 00 03 0F F7 0F 3F 1F 07 : 7D
F508 00 00 A0 A0 00 00 00 00 : 40
F510 C0 60 F0 E0 00 00 00 00 : F0
F518 20 90 F8 F0 E0 80 00 80 : 78
F520 00 00 01 03 0F 7F 3F 0F : E0
F528 00 00 01 03 0F 7F 3F 0F : E0
F530 00 01 02 0C 70 80 40 30 : 6F
F538 0C 43 FD F3 FF FD FD FF : 37
F540 0F 7C FE FF FF FF FF FF : 84
F548 73 BC 02 0C 00 03 03 03 : 46
F550 20 E0 9F FE C8 48 4A 5A : 51
F558 E0 7C FF FE EC CC CE FE : DD
F560 DC 3E E0 01 7E FE FF FF : 75
F568 00 00 00 00 00 00 00 00 : 00
F570 00 00 00 00 00 00 00 00 : 00
F578 00 00 80 00 00 00 00 00 : 80
---------------------------------
SUM: 4A 09 96 74 AD 4E F3 2D 0E1F

F580 01 00 00 00 00 00 00 00 : 01
F588 01 00 00 00 00 00 00 00 : 01
F590 0E 01 00 00 00 00 00 00 : 0F
F598 FF 07 3F 7F 7F 3F 3F 00 : C1
F5A0 00 00 07 0F 0E 0F 07 3F : 78
F5A8 01 F8 47 8F 8E 4F 47 C0 : B3
F5B0 58 C0 F0 F8 F8 F8 E8 28 : 00
F5B8 FC C0 30 18 08 00 B8 FC : C0
F5C0 FE FC 38 1C 0C 04 BC 3B : 55
F5C8 00 00 00 00 00 00 00 00 : 00
F5D0 00 00 00 00 00 00 00 00 : 00
```

```
F5D8 00 00 00 00 00 00 00 80 : 80
F5E0 00 00 00 00 00 00 00 00 : 00
F5E8 00 03 00 00 00 00 00 00 : 03
F5F0 03 04 07 00 00 00 00 00 : 0E
F5F8 00 00 02 02 00 00 00 00 : 04
---------------------------------
SUM: 64 83 EE 4B 27 99 E9 DE 1B1E

F600 FF FF 03 07 00 1F 1E 3F : 84
F608 00 00 FF 0F 1F 3F 3F 7F : 2A
F610 00 00 90 A0 00 00 00 00 : 30
F618 FF FF D3 B0 00 40 40 B8 : B9
F620 00 00 FC F8 F0 E0 F8 FC : B8
F628 00 00 00 00 00 00 00 00 : 00
F630 80 E0 00 00 00 00 00 00 : 60
F638 60 10 F0 00 00 00 00 00 : 60
F640 00 00 00 00 00 01 03 07 : 0B
F648 00 00 00 00 00 01 03 07 : 0B
F650 00 00 01 03 03 06 0C 18 : 31
F658 03 10 7F FC FF FF FF FF : 8A
F660 03 1F 7F FF FF FF FF FF : 9C
F668 1C 6F 80 03 00 00 00 00 : 0E
F670 08 F8 67 FF F2 52 52 56 : 52
F678 F8 1F BF FF FB F3 F3 FF : B5
---------------------------------
SUM: 00 A3 F6 5D FD C9 EA EB 5DA3

F680 F7 0F B8 00 1F FF FF FF : DA
F688 00 00 C0 80 00 00 80 80 : 40
F690 00 00 C0 80 00 00 80 80 : 40
F698 00 C0 20 40 80 80 C0 C0 : A0
F6A0 00 00 00 00 00 00 00 00 : 00
F6A8 00 00 00 00 00 00 00 00 : 00
F6B0 07 00 00 00 00 00 00 00 : 07
F6B8 FF 0F 0F 3F 7F 1F 1E 02 : 1A
F6C0 FF 0F 07 3E 78 1E 1D 0F : 15
F6C8 00 F0 37 7E F8 FE 3D 33 : 0B
F6D0 D6 90 FC FE FE FE BC C0 : D8
F6D8 FF B0 18 1A 0A 04 C0 FF : AE
F6E0 7F 7F 1A 1B 0B 05 C3 C0 : C6
F6E8 00 00 00 00 00 80 80 00 : 00
F6F0 00 00 00 00 00 C0 C0 00 : 80
F6F8 80 00 00 00 C0 E0 E0 C0 : C0
---------------------------------
SUM: D0 9C D3 6E 61 E1 96 42 910F

F700 00 00 00 00 00 00 00 00 : 00
F708 00 00 00 01 01 01 01 00 : 04
F710 00 01 03 03 03 03 03 03 : 13
F718 00 00 00 02 02 00 00 00 : 04
F720 3F FF 00 F7 F7 C0 80 00 : 6C
F728 C0 00 FF FF FF FF C0 80 : FC
F730 00 00 00 81 00 00 00 00 : 81
F738 FF FF 00 81 00 03 07 03 : 8C
F740 00 00 FF C3 83 07 0F 07 : 62
F748 00 00 A0 40 00 00 00 00 : E0
F750 C0 E0 F0 E0 00 C0 F0 FC : 1C
F758 20 10 F8 F0 E0 F0 FC FE : E2
---------------------------------
SUM: DE EF 89 D1 5F 7D 46 87 9CBC
```

リスト5 ソースリスト

```
0000             1 ;--------------------------------------
0000             2       ;     善司ソフト第2回作品
0000             3
0000             4       ;      WONDERERS FROM X1
0000             5 ;--------------------------------------
D000             6         ORG     0D000H
D000             7 YLEN:   EQU     16              ;画面縦のｷｬﾗｸﾀｰ幅
D000             8 WIDE:   EQU     80*YLEN         ;画面サイズ
D000             9 TEXT:   EQU     80*4+3000H      ;表示するﾃｷｽﾄ画面のｱﾄﾞﾚｽ
D000            10 CHR1:   EQU     0F400H          ;CHARACTER DATA ADR
D000            11 CHR2:   EQU     CHR1+120H       ;        "
D000            12 CHR3:   EQU     CHR2+120H       ;        "
D000            13 POSI:   EQU     80*16+38+4000H  ;CAHARACTERのﾎﾟｼﾞｼｮﾝ
D000            14 JMAX:   EQU     9               ;JUMP TIME
D000            15 STAY:   EQU     5               ;停空
D000            16 POSI2:  EQU     80*16+4000H  ;ﾄｹﾞのﾎﾟｼﾞｼｮﾝ
D000            17 COLD:
D000 CD 8A D2   18         CALL    PCG_ON          ;SCREEN INITIALIZE
D003            19
D003 CD A6 D2   20         CALL    ENMBUFCLR       ;ﾄｹﾞﾜｰｸ ｸﾘｱ
D006 21 26 45   21         LD      HL,POSI
D009 22 E2 D1   22         LD      (MYCHR),HL
D00C 3E 02      23         LD      A,2
D00E 32 E1 D1   24         LD      (JUMP),A        ;2=RUNNING 1=JUMP 0=DOWN
D011 3E 09      25         LD      A,JMAX
D013 32 E4 D1   26         LD      (MAX),A         ;JUMP のMAX
D016 21 E7 D1   27         LD      HL,ADRS
D019 22 E5 D1   28         LD      (PNT),HL
D01C            29 HOT:
D01C AF         30         XOR     A
D01D 32 EF D1   31         LD      (DEAD?),A       ;死んだかな?=0
D020            32 MAIN:
D020 CD BB D2   33         CALL    SCROLL          ;SCROLL
D023            34 K_IN:
D023 CD 49 D3   35         CALL    KEYIN
D026 FE 1B      36         CP      27              ;ESC
D028 CA C6 D3   37         JP      Z,PAUSE
D02B FE 20      38         CP      " "             ;SPACE KEY ?
D02D 20 30      39         JR      NZ,NON_SPC
D02F 3A E1 D1   40         LD      A,(JUMP)
D032 B7         41         OR      A
D033 28 2A      42         JR      Z,NON_SPC       ;落ちかけてるなら
D035 3A E4 D1   43         LD      A,(MAX)         ;ｽﾍﾟｰｽｷｰ押してもむだ.
D038 3D         44         DEC     A
D039 32 E4 D1   45         LD      (MAX),A
D03C 28 02      46         JR      Z,MADA          ;JUMP=0,つまりDOWN
D03E 3E 01      47         LD      A,1             ;JUMP=1,つまりJUMP
D040            48 MADA:
D040 32 E1 D1   49         LD      (JUMP),A
D043 3A E4 D1   50         LD      A,(MAX)
D046 FE 05      51         CP      STAY
D048 38 6A      52         JR      C,PNT_CK
D04A ED 4B E2   53         LD      BC,(MYCHR)
D04D D1
D04E CD F0 D1   54         CALL    CLR             ;古いｷｬﾗ消す
D051 2A E2 D1   55         LD      HL,(MYCHR)
D054 01 50 00   56         LD      BC,80
D057 B7         57         OR      A
D058 ED 42      58         SBC     HL,BC
D05A 22 E2 D1   59         LD      (MYCHR),HL
D05D 18 55      60         JR      PNT_CK
D05F            61
D05F            62 NON_SPC:
D05F 2A E2 D1   63         LD      HL,(MYCHR)
D062 E5         64         PUSH    HL
D063 3A E1 D1   65         LD      A,(JUMP)
D066 FE 02      66         CP      2
D068 20 1F      67         JR      NZ,OCHIRU?
D06A 3A 98 D3   68         LD      A,(KASCII)
D06D FE 34      69         CP      "4"
D06F 20 0B      70         JR      NZ,K6?
D071            71 JOGEN:  EQU     POSI-38+1
D071            72 JGN_H:  EQU     JOGEN/256
D071            73 JGN_L:  EQU     -JGN_H*256+JOGEN
D071 7D         74         LD      A,L
D072 D6 01      75         SUB     JGN_L
D074 7C         76         LD      A,H
D075 DE 45      77         SBC     JGN_H
D077 38 03      78         JR      C,K6?
D079            79                                 ;IF     HL>=POSI-38+1
D079            80
D079 2B         81         DEC     HL
D07A 18 0D      82         JR      OCHIRU?
D07C            83 K6?:
D07C FE 36      84         CP      "6"
D07E 20 09      85         JR      NZ,OCHIRU?
D080            86 KAGEN:  EQU     POSI+38
D080            87 KGN_H:  EQU     KAGEN/256
D080            88 KGN_L:  EQU     -KGN_H*256+KAGEN
D080 7D         89         LD      A,L
D081 D6 4C      90         SUB     KGN_L
D083 7C         91         LD      A,H
D084 DE 45      92         SBC     KGN_H
D086 30 01      93         JR      NC,OCHIRU?
D088            94                                 ;IF     HL<POSI+38      THEN
D088 23         95         INC     HL
D089            96 OCHIRU?:
D089 AF         97         XOR     A
D08A 32 E1 D1   98         LD      (JUMP),A
D08D 3E 09      99         LD      A,JMAX
D08F 32 E4 D1  100         LD      (MAX),A
D092 5D        101         LD      E,L
D093 54        102         LD      D,H
D094 01 50 00  103         LD      BC,80
```

▶「奥出雲の銘水シリーズ」には「しじみ汁」というのもあります。一度お試しください。え!? 味のほうは「二十世紀梨ドリンク」を飲める人だったら大丈夫。

小林 孝治 (25) 島根県

```
D097 09         104          ADD     HL,BC
D098            105 JIMEN:   EQU     POSI+40
D098            106 JMN_H:   EQU     JIMEN/256
D098            107 JMN_L:   EQU     -JMN_H*256+JIMEN
D098 7D         108          LD      A,L
D099 D6 4E      109          SUB     JMN_L
D09B 7C         110          LD      A,H
D09C DE 45      111          SBC     JMN_H
D09E 38 07      112          JR      C,J_CONT           ;地面についたか
D0A0            113                                     ;IF     HL>=POSI+40     THEN
D0A0 6B         114          LD      L,E
D0A1 62         115          LD      H,D
D0A2 3E 02      116          LD      A,2
D0A4 32 E1 D1   117          LD      (JUMP),A
D0A7            118 J_CONT:
D0A7 22 E2 D1   119          LD      (MYCHR),HL
D0AA C1         120          POP     BC
D0AB 7C         121          LD      A,H
D0AC B8         122          CP      B
D0AD 20 02      123          JR      NZ,CH_KS
D0AF 7D         124          LD      A,L
D0B0 B9         125          CP      C
D0B1            126 CH_KS:
D0B1 C4 F0 D1   127          CALL    NZ,CLR             ;IF     HL<>BC  THEN 古いのを消す
D0B4            128 PNT_CK:
D0B4 3A E1 D1   129          LD      A,(JUMP)
D0B7 FE 02      130          CP      2
D0B9 28 05      131          JR      Z,ANIM             ;ハシッテルナラ アニメスル
D0BB 21 40 F6   132          LD      HL,CHR3
D0BE 18 17      133          JR      WRTCH
D0C0            134 ANIM:
D0C0 2A E5 D1   135          LD      HL,(PNT)
D0C3 4E         136          LD      C,(HL)
D0C4 23         137          INC     HL
D0C5 46         138          LD      B,(HL)
D0C6 23         139          INC     HL
D0C7            140 AAMAX:   EQU     ADRS+8
D0C7            141 AM_H:    EQU     AAMAX/256
D0C7            142 AM_L:    EQU     -AM_H*256+AAMAX
D0C7 7D         143          LD      A,L
D0C8 D6 EF      144          SUB     AM_L
D0CA 7C         145          LD      A,H
D0CB DE D1      146          SBC     AM_H
D0CD 38 03      147          JR      C,PNTSET
D0CF            148                                     ;IF     HL>=ADRS+8 THEN
D0CF 21 E7 D1   149          LD      HL,ADRS
D0D2            150 PNTSET:
D0D2 22 E5 D1   151          LD      (PNT),HL
D0D5 69         152          LD      L,C
D0D6 60         153          LD      H,B
D0D7            154 WRTCH:
D0D7 ED 4B E2   155          LD      BC,(MYCHR)
D0DA D1
D0DB CD 16 D2   156          CALL    CHR_PUT
D0DE 3A EF D1   157          LD      A,(DEAD?)
D0E1 B7         158          OR      A
D0E2 CA 20 D0   159          JP      Z,MAIN
D0E5 C9         160          RET                        ;MISS and GAME OVER (RETURN TO BASIC)
D0E6            161
D0E6            162 ECH1:    EQU     49H
D0E6            163 ECH2:    EQU     59H
D0E6            164 COMES:   EQU     80*2+ENEMY_BUF+79
D0E6            165 JOI:     EQU     COMES/256
D0E6            166 KAI:     EQU     -JOI*256+COMES
D0E6            167 EDGE:    EQU     80*2+ENEMY_BUF
D0E6            168 EKZ:     EQU     8       ;剣の数
D0E6            169
D0E6            170 ENEMY:
D0E6 CD AA D1   171          CALL    SHIFT   ;仮想画面をSCROLL
D0E9            172
D0E9            173          ;ポインタを整理
D0E9 06 08      174          LD      B,EKZ
D0EB 21 C9 D1   175          LD      HL,TOGES
D0EE            176
D0EE            177 MOVE_LP:
D0EE 5E         178          LD      E,(HL)
D0EF 23         179          INC     HL
D0F0 56         180          LD      D,(HL)
D0F1 7A         181          LD      A,D
D0F2 B3         182          OR      E
D0F3 28 10      183          JR      Z,NEXT
D0F5 1B         184          DEC     DE                 ;トゲ X=X-1
D0F6            185 ED_H:    EQU     EDGE/256
D0F6            186 ED_L:    EQU     -ED_H*256+EDGE
D0F6 7B         187          LD      A,E
D0F7 D6 70      188          SUB     ED_L
D0F9 7A         189          LD      A,D
D0FA DE D4      190          SBC     ED_H
D0FC 30 03      191          JR      NC,WRT_AD
D0FE            192                                     ;IF     DE<EDGE THEN
D0FE 11 00 00   193          LD      DE,0               ;スクロールアウトしちゃったならワーク リセット
D101            194 WRT_AD:
D101 72         195          LD      (HL),D
D102 2B         196          DEC     HL
D103 73         197          LD      (HL),E
D104 23         198          INC     HL
D105            199
D105            200 NEXT:
D105 23         201          INC     HL
D106 10 E6      202          DJNZ    MOVE_LP
D108            203
D108 11 D9 D1   204          LD      DE,TMOVE
D10B 21 C9 D1   205          LD      HL,TOGES
D10E 3E 08      206          LD      A,EKZ
D110            207 EANM_LP:                            ;トゲ のアニメ
D110 08         208          EX      AF,AF'
D111 7E         209          LD      A,(HL)
D112 23         210          INC     HL
D113 46         211          LD      B,(HL)
D114 23         212          INC     HL
D115 E5         213          PUSH    HL
D116 6F         214          LD      L,A
D117 60         215          LD      H,B
D118 01 50 00   216          LD      BC,80
D11B B4         217          OR      H
D11C 28 45      218          JR      Z,NEXT_T           ;使われていなかった
D11E 1A         219          LD      A,(DE)
D11F B7         220          OR      A
D120 CA 58 D1   221          JP      Z,T0T1
D123 3D         222          DEC     A
D124 CA 48 D1   223          JP      Z,T1T2
D127 3D         224          DEC     A
D128 CA 38 D1   225          JP      Z,T2T3
D12B            226 ;T3T4                               ;アニメ
D12B 36 49      227          LD      (HL),ECH1
D12D CD 75 D1   228          CALL    ATARI?
D130 ED 42      229          SBC     HL,BC
D132 36 20      230          LD      (HL)," "
D134 AF         231          XOR     A
D135 C3 63 D1   232          JP      NEXT_T
D138            233
D138            234 T2T3:                               ;アニメ
D138 ED 42      235          SBC     HL,BC
D13A 36 49      236          LD      (HL),ECH1
D13C CD 75 D1   237          CALL    ATARI?
D13F ED 42      238          SBC     HL,BC
D141 36 20      239          LD      (HL)," "
D143 3E 03      240          LD      A,3
D145 C3 63 D1   241          JP      NEXT_T
D148            242 T1T2:                               ;アニメ
D148 ED 42      243          SBC     HL,BC
D14A 36 59      244          LD      (HL),ECH2
D14C ED 42      245          SBC     HL,BC
D14E 36 49      246          LD      (HL),ECH1
D150 CD 75 D1   247          CALL    ATARI?
D153 3E 02      248          LD      A,2
D155 C3 63 D1   249          JP      NEXT_T
D158            250 T0T1:                               ;アニメ
D158 36 59      251          LD      (HL),ECH2
D15A ED 42      252          SBC     HL,BC
D15C 36 49      253          LD      (HL),ECH1
D15E CD 75 D1   254          CALL    ATARI?
D161 3E 01      255          LD      A,1
D163            256 NEXT_T:                             ;ツギノトゲヘ
D163 12         257          LD      (DE),A
D164 13         258          INC     DE
D165 E1         259          POP     HL
D166 08         260          EX      AF,AF'
D167 3D         261          DEC     A
D168 C2 10 D1   262          JP      NZ,EANM_LP
D16B            263
D16B CD 9A D3   264          CALL    RND
D16E 7D         265          LD      A,L
D16F FE 14      266          CP      20
D171 DC 8B D1   267          CALL    C,AKI?             ;20/256以下なら新しいトゲが出るかもしんな
い
D174 C9         268          RET
D175            269
D175            270 ATARI?:
D175 C5         271          PUSH    BC
D176 E5         272          PUSH    HL
D177 01 30 71   273          LD      BC,-ENEMY_BUF+POSI2
D17A 09         274          ADD     HL,BC
D17B 7C         275          LD      A,H
D17C C6 80      276          ADD     A,80H
D17E 47         277          LD      B,A
D17F 4D         278          LD      C,L                ;主人公が
D180 ED 78      279          IN      A,(C)              ;とげにあたっているか
D182 28 03      280          JR      Z,NOT_HIT
D184 32 EF D1   281          LD      (DEAD?),A
D187            282 NOT_HIT:
D187 B7         283          OR      A
D188 E1         284          POP     HL
D189 C1         285          POP     BC
D18A C9         286          RET
D18B            287 AKI?:                               ;アタラシイトゲ シュツゲンカ?
D18B 21 C9 D1   288          LD      HL,TOGES
D18E 11 D9 D1   289          LD      DE,TMOVE           ;空いたワークをサーチ
D191 06 08      290          LD      B,EKZ
D193            291 AKI_LP:
D193 7E         292          LD      A,(HL)
D194 B7         293          OR      A
D195 28 06      294          JR      Z,KORENISURU
D197 23         295          INC     HL
D198 23         296          INC     HL
D199 13         297          INC     DE
D19A 10 F7      298          DJNZ    AKI_LP
D19C C9         299          RET
D19D            300
D19D            301 KORENISURU:
D19D 3E 49      302          LD      A,ECH1             ;イニシャルデータセット
D19F 32 BF D4   303          LD      (COMES),A
D1A2 36 BF      304          LD      (HL),KAI
D1A4 23         305          INC     HL
D1A5 36 D4      306          LD      (HL),JOI
D1A7 AF         307          XOR     A
D1A8 12         308          LD      (DE),A
D1A9 C9         309          RET
D1AA            310
D1AA            311 SHIFT:                              ;トゲ仮想画面をスクロール
D1AA 3E 20      312          LD      A," "
D1AC 11 D0 D3   313          LD      DE,ENEMY_BUF
D1AF 6B         314          LD      L,E
D1B0 62         315          LD      H,D
D1B1 23         316          INC     HL
D1B2 01 4F 00   317          LD      BC,79
D1B5 ED B0      318          LDIR
D1B7 12         319          LD      (DE),A
D1B8            320
D1B8 13         321          INC     DE
D1B9 23         322          INC     HL
D1BA 01 4F 00   323          LD      BC,79
D1BD ED B0      324          LDIR
D1BF 12         325          LD      (DE),A
D1C0            326
D1C0 13         327          INC     DE
D1C1 23         328          INC     HL
D1C2 01 4F 00   329          LD      BC,79
D1C5 ED B0      330          LDIR
D1C7 12         331          LD      (DE),A
D1C8 C9         332          RET
D1C9            333
D1C9            334 TOGES:
D1C9 00 00 00   335          DS      EKZ*2
D1CC 00 00 00
D1CF 00 00 00
D1D2 00 00 00
D1D5 00 00 00
D1D8 00
D1D9            336 TMOVE:
D1D9 00 00 00   337          DS      EKZ
D1DC 00 00 00
D1DF 00 00
D1E1 00         338 JUMP:    DS      1
D1E2 00 00      339 MYCHR:   DS      2
D1E4 00         340 MAX:     DS      1
D1E5 00 00      341 PNT:     DS      2
D1E7 00 F4 20   342 ADRS:    DW      CHR1,CHR2,CHR3,CHR2
D1EA F5 40 F6
D1ED 20 F5
D1EF 00         343 DEAD?:   DS      1
D1F0            344 CLR:                                ;キャラを消す
D1F0 11 03 04   345          LD      DE,0403H
D1F3            346 CLR_WRT:
D1F3 21 00 18   347          LD      HL,24*256          ;H=24,L=0
D1F6            348 CLR_LP0:
D1F6 ED 69      349          OUT     (C),L
D1F8 78         350          LD      A,B
D1F9 C6 08      351          ADD     A,8
D1FB 47         352          LD      B,A
D1FC 25         353          DEC     H
D1FD C2 F6 D1   354          JP      NZ,CLR_LP0
D200 CB F0      355          SET     6,B
D202 03         356          INC     BC
D203 15         357          DEC     D
D204 C2 F3 D1   358          JP      NZ,CLR_WRT
D207 16 04      359          LD      D,4
D209 69         360          LD      L,C
D20A 60         361          LD      H,B
D20B 01 4C 00   362          LD      BC,80-4
D20E 09         363          ADD     HL,BC
D20F 4D         364          LD      C,L
D210 44         365          LD      B,H
D211 1D         366          DEC     E
D212 C2 F3 D1   367          JP      NZ,CLR_WRT
D215 C9         368          RET
D216            369
D216            370 CHR_PUT:         ;(CHR SIZE は 4 × 3 固定)
D216            371          ; ENTRY
D216            372          ; HL = CHR_ADR
D216            373          ; BC = G_RAM ADR
D216            374          ;                          ;キャラクター表示ルーチン
D216 78         375          LD      A,B
D217 D6 10      376          SUB     10H
D219 59         377          LD      E,C
D21A 57         378          LD      D,A
D21B D5         379          PUSH    DE
D21C D9         380          EXX
D21D C1         381          POP     BC
D21E 11 03 04   382          LD      DE,403H            ;SIZE X=4/SIZE Y=3
D221 D9         383          EXX
D222            384 CHP_LP0:
D222 D9         385          EXX
D223 ED 78      386          IN      A,(C)
D225 D9         387          EXX
D226 FA 61 D2   388          JP      M,SPACE
D229 CD 53 D2   389          CALL    WRT
D22C            390
D22C            391 AD_CL:
D22C CB F0      392          SET     6,B
D22E 03         393          INC     BC
D22F D9         394          EXX
D230 03         395          INC     BC
D231 15         396          DEC     D
D232 D9         397          EXX
D233 C2 22 D2   398          JP      NZ,CHP_LP0
```

▶ファンタジーゾーンのウィンクロンの目の部分でうまい倒し方があります。ただ，アーケード版のやり方なので68版ではわかりませんが。まず，目玉が出て止まったら，左斜め下に位置しひたすら撃ちまくります。そうすると何発に1発かは目玉に当たるので，ねばればそのうち死んでくれます。まさに1ドットの狂いもダメ！　西本　英樹（17）北海道

```
D236 54        399          LD      D,H          ;ｼｸﾉｷﾞｮｳﾍGRAMｱﾄﾞﾚｽｦﾍﾝｺｳ
D237 5D        400          LD      E,L
D238 60        401          LD      H,B
D239 69        402          LD      L,C
D23A 01 4C 00  403          LD      BC,80-4
D23D 09        404          ADD     HL,BC
D23E 44        405          LD      B,H
D23F 4D        406          LD      C,L
D240 6B        407          LD      L,E
D241 62        408          LD      H,D
D242 D9        409          EXX
D243 16 04     410          LD      D,4
D245 60        411          LD      H,B
D246 69        412          LD      L,C
D247 01 4C 00  413          LD      BC,80-4
D24A 09        414          ADD     HL,BC
D24B 44        415          LD      B,H
D24C 4D        416          LD      C,L
D24D 1D        417          DEC     E
D24E D9        418          EXX
D24F C2 22 D2  419          JP      NZ,CHP_LP0
D252 C9        420          RET
D253           421
D253           422 WRT:                          ;ｼﾞｯｻｲﾆｶｷｺﾝﾃﾞｲﾙ ﾙｰﾁﾝ
D253 16 18     423          LD      D,8*3
D255           424 LP1:
D255 04        425          INC     B
D256 ED A3     426          OUTI
D258 78        427          LD      A,B
D259 C6 08     428          ADD     A,8
D25B 47        429          LD      B,A
D25C 15        430          DEC     D
D25D C2 55 D2  431          JP      NZ,LP1
D260 C9        432          RET
D261           433
D261           434 SPACE:
D261 11 00 18  435          LD      DE,24*256          ;E=0,D=24
D264           436
D264           437 SPC_LP:
D264 ED 59     438          OUT     (C),E
D266 23        439          INC     HL
D267 78        440          LD      A,B
D268 C6 08     441          ADD     A,8
D26A 47        442          LD      B,A
D26B 15        443          DEC     D
D26C C2 64 D2  444          JP      NZ,SPC_LP
D26F C3 2C D2  445          JP      AD_CL
D272           446
D272           447 BLCK:
D272 00 00 00  448          DB      0,0,0,0,0,0,0,0
D275 00 00 00
D278 00 00
D27A 00 00 00  449          DB      0,0,0,0,0,0,0,0
D27D 00 00 00
D280 00 00
D282 00 00 00  450          DB      0,0,0,0,0,0,0,0
D285 00 00 00
D288 00 00
D28A           451 PCG_ON:                       ;画面をＰＣＧ　ＯＮで埋める。
D28A 01 00 30  452          LD      BC,3000H
D28D 11 40 06  453          LD      DE,80*20
D290           454 PCG_LP:
D290 3E 20     455          LD      A,20H              ;SPACE
D292 ED 79     456          OUT     (C),A
D294 CB A0     457          RES     4,B
D296 3E 0F     458          LD      A,00100111B        ;COLOR 7 & PCG ON
NO LABEL ERR
D298 ED 79     459          OUT     (C),A
D29A CB E0     460          SET     4,B
D29C 03        461          INC     BC
D29D 1B        462          DEC     DE
D29E 7A        463          LD      A,D
D29F B3        464          OR      E
D2A0 20 EE     465          JR      NZ,PCG_LP
D2A2 C9        466          RET
D2A3           467
D2A3           468 ENMBUFCLR:                    ;ﾄｹﾞのﾜｰｸ初期化
D2A3 06 F0     469          LD      B,80*3
D2A5 21 D0 D3  470          LD      HL,ENEMY_BUF
D2A8           471 EC_LP:
D2A8 36 20     472          LD      (HL)," "
D2AA 23        473          INC     HL
D2AB 10 FB     474          DJNZ    EC_LP
D2AD 21 C9 D1  475          LD      HL,TOGES
D2B0           476
D2B0 06 18     477          LD      B,EKZ*2+EKZ
D2B2           478 EC_LP2:
D2B2 36 00     479          LD      (HL),0
D2B4 23        480          INC     HL
D2B5 10 FB     481          DJNZ    EC_LP2
D2B7 C9        482          RET
D2B8           483
D2B8           484 SCROLL:
D2B8           485
D2B8 11 00 E5  486          LD      DE,0E500H          ;ﾌﾟﾚｰﾝ 1
D2BB CD 19 D3  487          CALL    ROLL
D2BE 11 00 EA  488          LD      DE,0EA00H          ;ﾌﾟﾚｰﾝ 2
D2C1 CD 2D D3  489          CALL    ROLL2
D2C4           490          ;各プレーンの合成
D2C4 11 00 EF  491          LD      DE,0EF00H
D2C7 21 00 E0  492          LD      HL,0E000H          ;ﾌﾟﾚｰﾝ 0 (ﾌﾟﾚｰﾝ0は非ｽｸﾛｰﾙ)
D2CA 01 00 05  493          LD      BC,WIDE
D2CD ED B0     494          LDIR
D2CF 11 00 EF  495          LD      DE,0EF00H          ;仮想画面 ADDRESS
D2D2 21 00 E5  496          LD      HL,0E500H
D2D5 CD 07 D3  497          CALL    MIX                ;合成
D2D8           498
D2D8 CD E6 D0  499          CALL    ENEMY              ;障害物
D2DB 11 C0 F2  500          LD      DE,80*12+0EF00H
D2DE 21 D0 D3  501          LD      HL,ENEMY_BUF
D2E1 01 F0 00  502          LD      BC,3*80
D2E4 CD 0A D3  503          CALL    MIX2               ;合成
D2E7           504
D2E7 11 00 EF  505          LD      DE,0EF00H
D2EA 21 00 EA  506          LD      HL,0EA00H
D2ED CD 07 D3  507          CALL    MIX                ;合成
D2F0           508
D2F0 01 40 31  509          LD      BC,TEXT            ;VRAMに転送
D2F3 21 00 EF  510          LD      HL,0EF00H
D2F6 11 00 05  511          LD      DE,WIDE
D2F9           512 PRT_LP:
D2F9 04        513          INC     B
D2FA ED A3     514          OUTI
D2FC 03        515          INC     BC
D2FD 1B        516          DEC     DE
D2FE 7A        517          LD      A,D
D2FF B3        518          OR      E
D300 C2 F9 D2  519          JP      NZ,PRT_LP
D303 C9        520          RET
D304           521
D304           522 MIX:                          ;合成ﾙｰﾁﾝ
D304 01 00 05  523          LD      BC,WIDE
D307           524 MIX2:
D307           525 MIX_LP:
D307 7E        526          LD      A,(HL)
D308 FE 20     527          CP      " "                ;もしｽﾍﾟｰｽだったらなにもかかない
D30A 28 04     528          JR      Z,SKIP             ;つまり透明ということ
D30C 12        529          LD      (DE),A
D30D           530 SKIP:
D30D 23        531          INC     HL
D30E 13        532          INC     DE
D30F 0B        533          DEC     BC
D310 78        534          LD      A,B
D311 B1        535          OR      C
D312 C2 07 D3  536          JP      NZ,MIX_LP
D315 C9        537          RET
D316           538
D316           539 ROLL:                         ;仮想画面をｽｸﾛｰﾙ
D316 3E 10     540          LD      A,YLEN
D318           541 SCR_LOOP:
D318 08        542          EX      AF,AF'
D319 1A        543          LD      A,(DE)             ;左端を保存
D31A 6B        544          LD      L,E
D31B 62        545          LD      H,D
D31C 23        546          INC     HL
D31D 01 4F 00  547          LD      BC,79
D320 ED B0     548          LDIR
D322 12        549          LD      (DE),A             ;右端に左端を書く
D323 13        550          INC     DE
D324 08        551          EX      AF,AF'
D325 3D        552          DEC     A
D326 C2 18 D3  553          JP      NZ,SCR_LOOP
D329 C9        554          RET
D32A           655
D32A           556 ROLL2:                        ;仮想画面をｽｸﾛｰﾙ その2
D32A 3E 10     557          LD      A,YLEN
D32C           558 SCR_LOOP2:
D32C 08        559          EX      AF,AF'
D32D 13        560          INC     DE
D32E 1A        561          LD      A,(DE)
D32F F5        562          PUSH    AF
D330 1B        563          DEC     DE
D331 1A        564          LD      A,(DE)
D332 6B        565          LD      L,E
D333 62        566          LD      H,D
D334 23        567          INC     HL
D335 23        568          INC     HL
D336 01 4E 00  569          LD      BC,78
D339 ED B0     570          LDIR
D33B 12        571          LD      (DE),A
D33C F1        572          POP     AF
D33D 13        573          INC     DE
D33E 12        574          LD      (DE),A
D33F 13        575          INC     DE
D340 08        576          EX      AF,AF'
D341 3D        577          DEC     A
D342 C2 2C D3  578          JP      NZ,SCR_LOOP2
D345 C9        579          RET
D346           580
D346           581 KEYIN:                        ;キーインﾙｰﾁﾝ
D346           582    ;<  NONE
D346           583    ;>  A=ASCII CODE  (0 MEANS NO KEY)
D346           584    ;x  A
D346           585    ;-  BC DE HL
D346           586
D346 C5        587          PUSH    BC
D347 D5        588          PUSH    DE
D348 E5        589          PUSH    HL
D349           590
D349 3A 98 D3  591          LD      A,(KASCII)
D34C 32 99 D3  592          LD      (LASTKY),A
D34F FB        593          EI
D350 21 97 D3  594          LD      HL,KEYCON
D353 16 E6     595          LD      D,0E6H             ;D=&HE6 (COM)
D355 CD 71 D3  596          CALL    SEND1
D358 CD 83 D3  597          CALL    CANW
D35B F3        598          DI
D35C           599
D35C CD 7A D3  600          CALL    GET1
D35F 72        601          LD      (HL),D
D360 23        602          INC     HL
D361 CD 7A D3  603          CALL    GET1
D364 72        604          LD      (HL),D
D365 23        605          INC     HL
D366 FB        606          EI
D367           607
D367           608    ;HL=LASTKY ﾉ ADDRESS ﾆ ﾅｯﾃｲﾙ.
D367           609
D367 3A 98 D3  610          LD      A,(KASCII)
D36A E1        611          POP     HL
D36B D1        612          POP     DE
D36C C1        613          POP     BC
D36D C9        614          RET
D36E           615
D36E           616 SEND1
D36E CD 83 D3  617          CALL    CANW
D371 01 00 19  618          LD      BC,01900H
D374 ED 51     619          OUT     (C),D
D376 C9        620          RET
D377           621
D377           622 GET1
D377 CD 8D D3  623          CALL    CANR
D37A 01 00 19  624          LD      BC,01900H
D37D ED 50     625          IN      D,(C)
D37F C9        626          RET
D380           627
D380           628 CANW
D380 01 01 1A  629          LD      BC,01A01H
D383           630 CANWL
D383 ED 78     631          IN      A,(C)
D385 E6 40     632          AND     040H
D387 20 FA     633          JR      NZ,CANWL
D389 C9        634          RET
D38A           635
D38A           636 CANR
D38A 01 01 1A  637          LD      BC,01A01H
D38D           638 CANRL
D38D ED 78     639          IN      A,(C)
D38F E6 20     640          AND     020H
D391 20 FA     641          JR      NZ,CANRL
D393 C9        642          RET
D394           643
D394 00        644 KEYCON:          DS 1
D395 00        645 KASCII:          DS 1
D396 00        646 LASTKY:          DS 1
D397           647
D397           648 RND:
D397           649          ;       乱数ﾙｰﾁﾝ
D397           650          ;ENTRY: x
D397           651          ; OUT : HL=0-65535
D397           652
D397 ED 5B AE  653   LD  DE,(OLDRND)
D39A D3
D39B ED 4B B1  654   LD  BC,(STEP)
D39E D3
D39F CD B3 D3  655   CALL  MULTI
D3A2 22 AE D3  656   LD  (OLDRND),HL
D3A5 ED 5F     657   LD  A,R
D3A7 32 B0 D3  658   LD  (REFR),A
D3AA C9        659   RET
D3AB           660
D3AB 23 E1     661 OLDRND: DW      0E123H
D3AD 00        662 REFR:   DS      1
D3AE 83 03     663 STEP:   DW      899
D3B0           664
D3B0           665 MULTI
D3B0 21 00 00  666   LD  HL,0
D3B3 3E 10     667   LD  A,10H
D3B5           668 MLOOP:
D3B5 29        669   ADD HL,HL
D3B6 CB 23     670   SLA E
D3B8 CB 12     671   RL  D
D3BA D2 C1 D3  672   JP  NC,RSKIP
D3BD 09        673   ADD HL,BC
D3BE           674 RSKIP
D3BE 3D        675   DEC A
D3BF C2 B5 D3  676   JP  NZ,MLOOP
D3C2 C9        677   RET
D3C3           678
D3C3           679 PAUSE:
D3C3 CD 46 D3  680          CALL    KEYIN
D3C6 FE 1B     681          CP      27                 ;ESCが押されている間ﾙｰﾌﾟ
D3C8 28 F9     682          JR      Z,PAUSE
D3CA C3 23 D0  683          JP      K_IN
D3CD           684
D3CD           685 ENEMY_BUF:
D3CD 00        686          DS      1         ;本当は80*3個
D3CE           687
D3CD [D3CD]
```

MZ-2500用グラフィックエディタ作成講座〈最終回〉

完成！ 画餅システム

Motohashi Jun
本橋　純

今回でいよいよ，グラフィックエディタ作成も終わりです。絵（画）に描いた餅にはならず，予告したスペックどおり堂堂の完成です。本橋氏の実力が発揮されましたね。MZ-2500ユーザー以外の方も十分に楽しんでいただけたと思います。

最後の仕上げだ

さて，最後を飾るのは印刷ウィンドウと諸設定ウィンドウです。あらかじめ断っておきますが，印刷機能の対応プリンタは我らがMZ-1P17系とNEC PC-PR406の2種類です。うぉぉぉぉ，俺のプリンタには対応していないのかぁぁぁと憤ってしまった人もご安心を。なるべく多くの機種で対応できるように工夫してありますから。それと，諸設定ウィンドウではこれまで封印されていた透明色やマウスカーソルの色変更などの機能が解放されます。それでは始まり始まりー。

機能解説

1)　諸設定ウィンドウ

Mouse……マウスカーソルの色を表示してある部分でクリックすると現在のペンカラーがカーソルの色になります。なぜ，表示している色とカーソルの色が違うかというと，8プレーンのうちの半分の4プレーンで表示させているからです。これは速度的な問題とG-RAMデータを保存しておくだけの余裕がないというのが理由です。自分で使っていても少々見づらいかなと反省しています。

Cblock……BASICでのcblock文と同じです。ここで反転していない要素が4階調表示になります。

Rope……閉曲線指定の際のラインの色を指定します。

Mask……マスキングの色を指定します。

透明色……色の位置で左クリックするとその色が透明色として指定されます。スイッチでこの機能を使用するか否かを選択できます。

SPDレバー……マウスの移動速度を調整できます。デフォルトは最高速ですからこの速度に慣れるとほかのグラフィックエディタが使いにくくなるかもしれません。まあ，光学式マウスで絵を描くときの使いづらさに比べればたいした問題ではないでしょうが……。

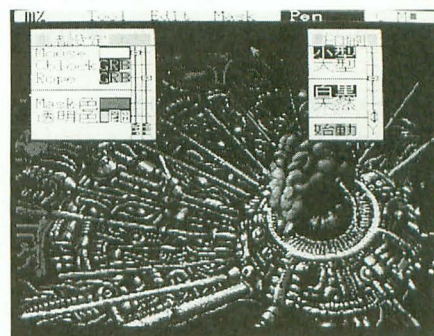
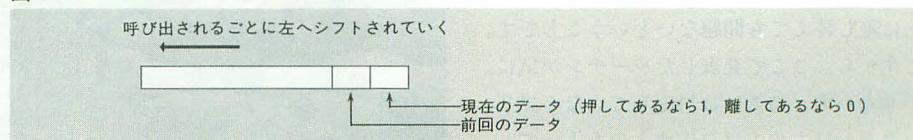
DBLレバー……ダブルクリックの間隔を調整できます。デフォルトは0.2秒で，設定範囲は0.1～0.5秒となっています。

2)　印刷ウィンドウ

画面をプリンタでハードコピーします。サイズは大型，小型の2種類あるので好きなほうを選べます。で，色もモノクロかカラーかを選択できます。そして，右側についているレバーで縦/横の比率を指定します。この比率のデフォルトは1.2で，設定範囲は1.0～1.8の間です。

以上のパラメータを設定してから，始動の位置で左クリックし，マウスで印字範囲を指定してください。これで印字が開始されます。

なお，プリンタの電源を入れてない場合や受信状態になっていない場合は始動をクリックしてもなにもしませんので注意してください。プリンタがReadyになるまで待待って，しばらくしてから“LPT not ready error”が出るような極悪な対応はいやでしたので，わざわざI/Oポートでプリンタの状態を調べているわけです。

図1

これが最後のウィンドウだ

解説：諸設定ウィンドウ

ここではダブルクリックの間隔をどのようにして測定しているかについてお話ししておきます。操作方法のところで0.2秒という具体的な数値が出ていたので不思議に思った人もいるのではないでしょうか。

BASICにはTIME関数がありますね。これを利用していると考えてもらえば，ほぼ間違いありません。メモリの0640Hからはテーブルソフトウェアタイマとなっています。データ形式は，

第0バイト　0で停止，1でカウント
第1～3バイト　24ビットカウンタ
（0.1秒ごとにインクリメントされる）
第4～7バイト　不明

の8バイトで構成されています（第4バイト以降は使っていないので調べていません)。これが8個分並んでおり，最初の4つはFDCやTIME関数，input wait, pause, on interval call などに使用されています。残

りの4つはどうやら空いているらしいので，ここを左右クリックの2つ分使用しています。

当初はここを使わずに処理するようにしていたのですが（マウスルーチンが呼ばれるたびにカウントする），どうも間隔にムラが出てしまうのでうまくダブルクリックを判定してくれません。ですから，正確なタイマを使ったのです。

えっ？　ダブルクリックの判定方法がウヤムヤのままじゃないかって？　それじゃあ，ついでにマウス移動ルーチンも軽く解説しておきましょうか（うーん，強引な展開)。

マウスの位置やボタンが押されたかどうかの判定はIOCSコールを利用することで得られます。で，処理ルーチンが呼び出されるたびに以下に述べるような処理を行います。

まず，前回呼び出されたときのマウス位置をワークに入れ，現在位置をIOCSコールで検出しワークに格納します。このワークエリアは各ボタンとも図1のようになっています。これからわかるとおり，単なるクリックの場合は前回1で今回0だから，

((データ) and 3)

が2か否かで判定できますね。

ダブルクリックは（シングル）クリックしたあとにカウント開始するようにします。そうして設定時間内にもう1回クリックされたならダブルクリックされたものとして確認され，制限時間を超えていたならシングルクリックが行われたものとして認識されます。詳しくはソースリストを参照してください。

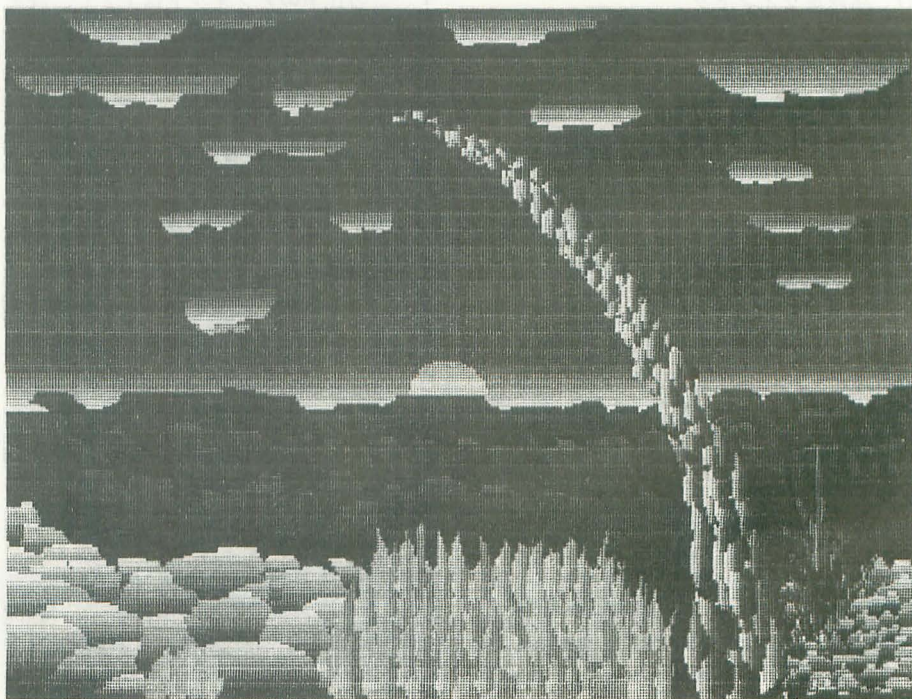
解説：印刷

最初に述べたとおり，ハードコピールーチンはMZ-1P17系とPC-PR406系にしか対応していません。なお，このハードコピールーチンはスタンドアロンに作ってありますので，このルーチン単体でも動作可能です。いい換えれば，ワークエリアと先頭アドレスさえあわせれば自分で作ったルーチンに差し替えても問題ないということです。ですから，ここで発表したルーチンが気にいらない人，あるいは対応していないプリンタをお持ちの方は自作するのもよいでしょう。

ここでのハードコピールーチンは画面上の1ピクセルを2×2，あるいは4×4ドットで構成して打ち出す方式を採用しています。ディザ法やら誤差拡散法などといった高度なことは一切していません（だから満足できないときは自作してくれといったわけね)。まあ，ゼロの状態から愛用のプリンタ専用ルーチン完成まで3日しかかかっていないので，高度なことはしてなくても当然といえば当然ですね。もう，うすうすおわかりとは思いますが，私の愛用プリンタは日電のものです。

さて，それではルーチンの解説に入りますか。いつかのトランスフォームルーチンのときとは逆にトップダウンに述べていきましょう（余談だが，私は最近トップダウンに設計することのほうが多くなっている)。

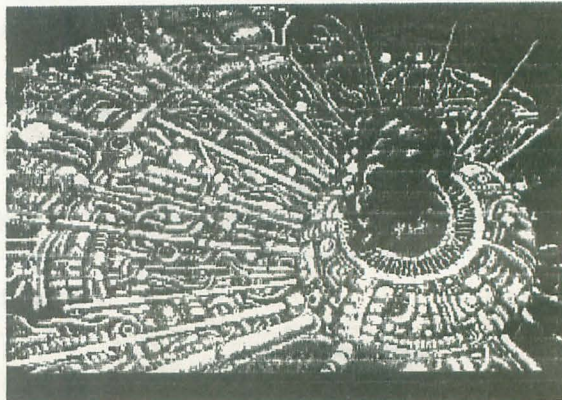
0)　全体的流れ

画面上で座標（X, Y）を左上端として幅lx，高さlyドットの範囲を考えます。ここでは比率（縦/横）を4/3にし（lpsst＝256＊3/4)，24ピンプリンタを用いて（lppin＝24)，小型モード（1ピクセル＝2×2ドット）で印刷することにしましょう。

ハードコピーはヘッドが移動して印刷しラインフィードをする，といったことを，（比率×ly/lppin）回繰り返せばいいことになります。24ピン，小型モードなら画面上で縦（lppin/lpdot)＝12ドット分を1回のヘッド移動で印刷するわけです。プリンタ側から見ると（ly×lpdot）ドット分となります。

まとめると，24ピン＝24ドット＝3バイトを横（lx×lpdot）ドット分，つまり，1回のヘッド移動で(lppin/8)×lx×lpdot バイト分のビットイメージデータをプリンタ側に送ることになるわけです。

ビットイメージデータの並びは24ピンの場合なら図2のようになります。小型モードの場合なら0,3→1,4→2,5→6,9……と画面上の1ピクセルに対応した2×2ドット分のイメージデータを設定していくことに

実行例　小型モード（縮小率82%）

実行例　大型モード（縮小率66%）

なります。ここで注意しなければならないのは，プリンタによってイメージデータのビットごとの順序が異なっているということです。図３を参照してください。

1)　モノクロモード

まず，画面上のドットデータは図４のような形式でプリンタヘッドのピンに対応したドット数だけ読み込んでしまいます。このときはスタンドアロンですから画餅の自作ルーチンは使いません。ですから，IOCSコールのSVC XXHでRGB輝度に変換します。このあたりのスピードはBASICでやるのと同じなので結果としてルーチン全体が遅くなってしまいました。で，RGB別にしたデータを階調表示で印刷します。階調は小型サイズが８階調，大型が16階調で表現されるようにしています。

ちなみに，階調ごとのマトリクスデータは私の独断と偏見で作ってあるのでこれが気にいらない人は例によって，自分で改造してください。なお，輝度の変換式はOh! X '88年11月号にあった。

輝度＝0.3＊R＋0.59＊G＋0.11＊B

という式を使っています。

2)　天然色モード

カラーの場合はRGB別データをプリンタのインクリボンの色要素に変更しなければなりません。その要素がPC-PR406の場合は黒(K)，マゼンタ(M)，シアン(C)，黄(Y)の４色で，MZ-1P17の場合は青(B)，マゼンタ(M)，黄(Y)の３色です。

で，実際の色変換は図５のようになります。その表からわかるとおり，PC-PR406ならRGBそれぞれNOTをとることでCMYとなり，特にRGBが黒のときは黒のリボンも使ってやればよいわけです。これを輝度４，２，１のときそれぞれの場合について変換し，再びまとめるとCMYの輝度になります。

ところが，MZ-1P17の場合はシアンではなく，青といったわけのわからん構成になっているので図５で示したような変換式になってしまいます（わざわざカルノー図で求めたわけだ，こりゃ)。これだと黄や青はともかく，マゼンタの式はNOTやらANDやらORがありすぎて，こいつぁ面倒だということになります。そこで実際の処理では安直に表そのものを用いることにしたわけです。

こうして求まったリボン用輝度（この場合は濃淡といったほうがいいのかな）からモノクロモードと同様にパターンで印刷するわけです。なお，大型サイズでは２倍した値のパターンを使います。

3)　非対応プリンタのための変更点

例によってソースリストのなかにくどいほど注釈を記入しておきました。

カラーのときは2)の説明からわかるように，メーカーによっていろいろと仕様が異なっていて困りものです。しかし，モノクロモードならなんとかなりそうです。ここではプログラム中のデータ設定について述べておきましょう。まず，PC-PR406，MZ-1P17の設定手順を図６に示しておきます。これと根本的に異なる場合は部分的な改造が必要です。

・ピン数（lppin）は8，16，24のいずれかです。48ピンには残念ながら対応していません。

・プリンタモード（printe）は７ビットを１とすると図３の(1)，０とするなら(2)のプリントイメージの順序となります。そこで，０ビットが１ならMZ-1P17，１ならPC-PR406といった手順で印刷していきます。

・コピーモード（LCOPYM）はコピー用のプリンタの設定コードが入っています。必要なときは適当なコード（０でよい）をダミーとして入れておいてください。

・ラインフィード幅（LLFDAT）に設定コ

図２

図３

図４

図５

G	R	B	M	C	Y	K
0	0	0	1	1	1	1
0	0	1	1	1	0	0
0	1	0	1	0	1	0
0	1	1	1	0	0	0
1	0	0	0	1	1	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	0

G	R	B	Y	M	B
0	0	0	1	1	1
0	0	1	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
1	0	0	1	0	1
1	0	1	0	1	1
1	1	0	1	0	0
1	1	1	0	0	0

406の場合

$M=\bar{G}$

$C=\bar{R}$

$Y=\bar{B}$

$K=\bar{G}\bar{R}\bar{B}$

1P17の場合

$M=R\bar{G}+\bar{B}\bar{G}+B\bar{R}G$

$B=\bar{R}$

$Y=\bar{B}$

ードを入れておきます。

・印字指令(CR)，ラインフィード指令（LF)，ビットイメージ指令(LBITIM)，カラー指令(LCOLDT)それぞれを設定します。

終わりにかえて

画餅は機能ごとに入力するということで次々にパワーアップされるようになっていますが，もともと単発で掲載されることを考慮して設計したものが連載というかたちをとるようになったにすぎません（でなきゃ，パッチ当てもせず組み込めたりほとんど隙間なくプログラムが詰まっているわけないでしょ)。

単発というからには1回ですべてが掲載されるわけで，当然リストも巨大なものになってしまいます。そこで少しでも入力が楽になるように，最小限のプログラムさえ打ち込めば動作するような設計にしておいたわけです。このことでプログラム自体も上下関係が明確にでき，スッキリとした構造にできました。この設計はなかなかおいしかったと思います。

さて，この連載はグラフィックエディタ作成講座ということでグラフィック関係に必要な処理，アルゴリズムを解説してきましたが，それぞれは結構単純な処理で実現できていたのがおわかりと思います（手抜きをしたから単純という説もある)。欲張った機能でオーバースペック気味の画餅でさえ中身はこんなもんなんです。まあ，中身はどうあれ完成させてしまえばこっちのもんですから。

それでは，これまでこの連載を読んで，プログラムを入力してくださった方々に感謝しつつここで筆を置くことにしましょう。この連載がなんらかのかたちでお役に立てば幸いです。

図6

(MZ-1P17)

改行幅設定	ESC＋“%6”＋16	
カラーモードなら	ESC＋19_Hを設定	
ビットイメージ	ESC＋“%1”＋H＋L	HLで16ビットの数値で印字する横ドット数
カラーモードなら	ビットイメージ指令が送られるごとに Y，M，Bのリボンで印字	
印字指令	$0D_H$	
改行指令	$0A_H$	

(PC-PR406)

コピーモード設定	ESC＋“D”	
改行幅設定	ESC＋“T”＋“18”	
カラーモードなら	ESC＋“C”＋カラーコード でリボンの指定	
ビットイメージ設定	ESC＋“J”＋“$d_3d_2d_1d_0$”	d_3, d_2, d_1, d_0 は数字のASCIIコードで，4桁の自然数を表し印字する横方向のドット数である。
印字指令	$0D_H$	
改行指令	$0A_H$	

リスト1 諸設定ウィンドウ

```
F2A6 09 00 01 01 0B 01 3F B3 : 09
F2AE F4 F3 00 07 02 09 02 50 : 4B
F2B6 F4 FA F3 00 07 03 09 03 : F7
F2BE D3 F4 2D F4 00 07 04 09 : FC
F2C6 04 E5 F4 34 F4 00 07 06 : 12
F2CE 09 06 56 F4 FF F3 00 07 : 52
F2D6 07 07 07 83 F4 0F F4 00 : 8F
F2DE 08 07 09 07 FF F4 41 F4 : 47
F2E6 00 0A 02 0A 07 9C F4 18 : C5
F2EE F4 00 0B 02 0B 07 AE F4 : B5
F2F6 1F F4 03 77 05 05 61 49 : 41
F2FE 03 75 50 40 5F 47 02 07 : B7
F306 04 04 63 4B 02 70 04 04 : 30
F30E 62 4A 01 70 04 2B 50 2B : C7
F316 02 70 50 0F 58 4A 01 70 : E4
F31E 05 0F 61 0F 03 07 38 10 : D6
-----------------------------------
SUM: 63 1A F0 4A D1 E5 1C 1B 03D2

F326 4F 17 03 74 38 18 3F 26 : 92
F32E 03 72 40 18 47 26 03 71 : AE
F336 48 18 4F 26 01 77 38 1F : A4
F33E 4F 1F 02 07 38 30 4F 37 : 65
F346 02 07 38 38 3F 3F 0B 06 : 08
F34E 00 07 10 08 01 01 09 24 : 4E
F356 20 20 20 20 20 20 24 24 : 08
F35E 24 24 00 08 0A 02 09 36 : 9B
F366 36 1F 1D 1D 36 36 1F 1D : 37
F36E 1D 36 36 1F 1D 1D 36 36 : 4E
F376 00 08 0A 06 09 36 36 1F : AC
F37E 1D 1D 36 36 00 08 0A 08 : C0
F386 09 38 3A 00 08 08 07 09 : 9B
F38E 3E 3C 00 07 40 08 07 03 : D3
F396 09 47 52 42 00 07 00 08 : F3
F39E 02 01 09 8F 94 90 DD 92 : 2E
-----------------------------------
SUM: F1 48 24 71 5A 7F 8A 91 B73D

F3A6 E8 00 08 01 02 09 4D 6F : B8
F3AE 75 73 65 00 08 01 03 09 : 62
F3B6 43 62 6C 6F 63 6B 00 08 : 56
F3BE 01 04 09 52 6F 70 65 20 : C4
F3C6 20 47 52 42 00 08 01 06 : 0A
F3CE 09 4D 61 73 6B 90 46 00 : 6B
F3D6 08 01 07 09 93 A7 96 BE : A7
F3DE 90 46 00 00 01 02 00 00 : D9
F3E6 02 01 00 01 02 00 01 02 : 09
F3EE 01 00 02 01 02 00 11 F8 : 0F
F3F6 F2 C3 3A AE 21 B2 F9 18 : 81
F3FE 03 21 51 FA 4E 11 15 05 : E8
F406 06 00 21 01 01 7D C3 BF : 28
F40E BD 3A 53 FA 11 05 05 4F : AE
F416 18 EE 3A 9E F9 06 0F 18 : 04
F41E 05 3A B0 F9 06 08 4F 78 : BD
-----------------------------------
SUM: 3A FB 87 BC 5F 79 D8 19 2398

F426 91 4F 3E 01 C3 5F BC 3A : 37
F42E 54 FA 06 00 18 12 3A 52 : 0A
F436 FA 4F 3E 24 91 CB 3F 06 : 4C
F43E 01 18 05 3A 57 FA 06 01 : B0
F446 4F 78 11 01 01 06 00 C3 : A3
F44E A5 BC 21 B2 F9 C3 59 F4 : 3D
F456 21 51 FA CD BB A4 C0 E5 : 3D
F45E CD 8C A2 E1 3A 38 FA 77 : BF
F466 AF 32 56 FA CD 97 A4 11 : 4A
F46E 15 05 3A 38 FA 4F 21 01 : F7
F476 01 06 00 7D CD BF BD CD : 9A
F47E F3 B9 C3 C8 A4 CD BB A4 : 07
F486 C0 CD 8C A2 3A 38 FA 32 : 59
F48E 53 FA AF 32 56 FA CD 97 : E2
F496 A4 11 05 05 18 D4 21 9E : 6A
F49E F9 06 0F CD B3 F4 D0 21 : 73
-----------------------------------
SUM: 2A 95 F7 DD 45 47 43 B1 0EEC

F4A6 9E F9 7E 23 77 C3 7A A3 : 8F
F4AE 21 B0 F9 06 08 3A AC F9 : B7
F4B6 0F D0 E5 C5 CD 8C A2 C1 : 45
F4BE E1 78 F5 96 4F E5 AF CD : 94
F4C6 5F BC E1 F1 91 77 3E 01 : 34
F4CE CD 5F BC 37 C9 3A 54 FA : 70
F4D6 06 01 CD 11 F5 C0 32 54 : 20
F4DE FA CD 2D F4 C3 27 F5 3A : 01
F4E6 52 FA 4F 3E 24 91 CB 3F : 98
F4EE 06 00 CD 11 F5 C0 87 4F : 6F
F4F6 3E 24 91 32 52 FA C3 34 : 68
F4FE F4 3A 57 FA 06 00 CD 11 : 63
F506 F5 C0 32 57 FA CD 41 F4 : 3A
F50E C3 8E B9 4F CD BB A4 C0 : 45
F516 C5 CD 8C A2 C1 11 01 01 : 94
F51E 78 06 00 CD A5 BC AF 79 : D4
-----------------------------------
SUM: 5A 53 63 41 4B A6 A7 B4 7E62

F526 C9 3A 54 FA 4F 87 81 5F : 07
F52E 16 00 79 D5 21 E2 F3 19 : 73
F536 11 A6 FA 01 03 00 ED B0 : 52
F53E D1 21 EB F3 19 11 A9 FA : 9D
F546 01 03 00 ED B0 FE 02 3F : E0
F54E 38 07 B7 3E 31 28 02 3E : CD
F556 0D DF 5D 3A AD CF B7 C4 : 7A
F55E 0C D1 C9                : A6
-----------------------------------
SUM: 13 BB 8F 28 1A 6F C5 63 94FD
```

リスト2 印刷ウィンドウ

```
F561 05 00 01 01 06 01 3F B3 : 00
F569 2E F6 00 01 02 04 03 4F : 7D
F571 F6 34 F6 00 01 05 04 06 : 30
F579 54 F6 39 F6 00 06 02 06 : 87
F581 06 75 F6 46 F6 00 01 08 : B6
F589 04 08 8D F6 00 00 03 77 : 09
F591 05 05 39 49 02 07 04 04 : 9D
F599 3B 4B 02 70 04 04 3A 4A : 84
F5A1 02 70 04 24 2F 3C 01 70 : 76
F5A9 05 0F 39 0F 03 75 30 38 : 3C
F5B1 37 3F 03 72 08 40 27 47 : A1
F5B9 03 76 08 10 27 1F 03 76 : 50
F5C1 08 28 27 37 02 70 2F 10 : 3F
F5C9 2F 48 0B 06 00 07 10 08 : A7
F5D1 01 01 09 24 20 20 20 20 : AF
F5D9 24 00 08 06 02 09 36 1F : 92
-----------------------------------
SUM: 64 92 79 09 8A CB 7A 97 86F8
F5E1 1D 36 1F 1D 36 00 08 06 : D3
F5E9 05 09 36 1F 1D 36 1F 1D : F2
F5F1 28 00 07 00 08 02 01 09 : 43
F5F9 88 F3 8D FC 00 08 01 02 : 0F
F601 09 8F AC 8C 5E 00 08 01 : 37
F609 03 09 91 E5 8C 5E 00 08 : 74
F611 01 05 09 94 92 8D 95 00 : 57
F619 08 01 06 09 93 56 91 52 : E4
F621 00 08 01 08 09 8E 6E 93 : A9
```

```
F629 AE 20 59 00 00 11 8F F5 : BC
F631 C3 3A AE 3A 66 FB 18 03 : 61
F639 3A 67 FB 47 0E 00 11 04 : 06
F641 01 7A C3 A5 BC 3A 68 FB : 3C
F649 4F 3E 01 C3 5F BC 21 66 : F3
F651 FB 18 05 21 67 FB 18 00 : B3
F659 CD BB A4 C0 E5 CD 8C A2 : CC
---------------------------------
SUM: AA 24 A5 18 4E D9 AA 1B 49E7

F661 E1 11 04 01 46 AF 4F D5 : 10
F669 E5 CD A5 BC E1 D1 70 3E : 73
F671 01 C3 A5 BC 3A AC F9 0F : 13
F679 D0 E5 CD 8C A2 E1 3A 68 : 33
F681 FB 4F AF CD 5F BC 79 32 : 8C
F689 68 FB 18 B9 CD BB A4 C0 : 20
F691 DB FE 0F D8 0F D0 CD 10 : 7C
F699 BC CD CD B8 CD 59 C0 C0 : B4
F6A1 CD 7C A8 38 F7 CD 02 A9 : 98
F6A9 AF CD 97 A4 3E 02 CD 3D : 01
F6B1 BF F5 3E 07 CD 28 BF 3A : E7
F6B9 5E F9 CD 32 BF 2A 79 F9 : B1
F6C1 22 03 5B 2A 7B F9 22 05 : 45
F6C9 5B 2A 81 F9 23 22 07 5B : A6
F6D1 2A 83 F9 23 22 09 5B 3A : 89
F6D9 66 FB 3C 87 32 0C 5B 2E : EB
---------------------------------
SUM: 37 7D 19 FD BE FE 82 2D FF63

F6E1 18 3A 68 FB D6 03 67 CD : C2
F6E9 00 BF 65 25 22 0D 5B 3A : 0D
F6F1 67 FB 32 0F 5B CD 00 5B : 26
F6F9 F1 CD 28 BF C3 B2 C0 00 : DA
---------------------------------
SUM: 70 C1 27 EE 16 8F 82 62 017C
```

リスト3 ハードコピールーチン

```
FB00 C3 31 5B 00 00 00 00 00 : 4F
FB08 00 00 00 18 00 00 00 00 : 18
FB10 00 0A 00 00 00 00 00 1B : 25
FB18 25 36 10 00 00 0A 00 00 : 75
FB20 00 0D 00 00 00 1B 25 31 : 7E
FB28 00 00 00 1B 19 00 00 00 : 34
FB30 00 CD 38 5B CD E1 5B C9 : 32
FB38 3A 0B 5B CB 3F 5F 3A 0C : 4F
FB40 5B FE 02 20 02 CB 3B 7B : FE
FB48 ED 5B 07 5B 21 00 00 CD : 98
FB50 44 01 22 F3 5E 2A 07 5B : 44
FB58 29 3A 0C 5B FE 02 28 01 : F3
FB60 29 22 F5 5E 3A 10 5B CB : 0E
FB68 47 28 19 06 04 11 0A 00 : AD
FB70 CD 53 01 7B C6 30 F5 10 : 97
FB78 F4 21 F7 5E 06 04 F1 77 : DC
---------------------------------
SUM: 08 A8 3B 5F AE B1 6F 17 EE05

FB80 23 10 FB 70 21 11 5B CD : F8
FB88 7D 5E 21 17 5B CD 7D 5E : 16
FB90 2A 03 5B 22 E8 5E 2A 05 : 1F
FB98 5B 22 EA 5E AF 32 E6 5E : EA
FBA0 2A 0D 5B 65 25 22 0D 5B : A6
FBA8 3A 0D 5B 5F 16 00 3A 09 : 5A
FBB0 5B 67 6A CD 44 01 3A 0C : 84
FBB8 5B FE 02 3A 0B 5B 0E 08 : 11
FBC0 28 04 CB 3F CB 39 CB 3F : 44
FBC8 CB 39 32 EE 5E 5F 79 32 : 8C
FBD0 E7 5E 6C 26 00 54 CD 53 : 4B
FBD8 01 7B B7 28 01 2C 45 04 : D1
FBE0 C9 C5 3A 0F 5B B7 28 05 : 16
FBE8 CD FA 5B 18 03 CD 3D 5C : A3
FBF0 2A F0 5E 22 EA 5E C1 10 : B3
FBF8 E8 C9 CD 64 5C D5 3E 04 : 55
---------------------------------
SUM: C2 A0 63 FA 6B BB 31 43 F49E

FC00 93 32 E6 5E CD 74 5C 2A : D0
FC08 03 5B 22 E8 5E DD 21 00 : C4
FC10 80 ED 4B 07 5B C5 CD 10 : BC
FC18 5E CD CE 5C 2A E8 5E 23 : E8
FC20 22 E8 5E C1 0B 78 B1 20 : 7D
FC28 EC CD 99 5C D1 1D 20 CD : 89
FC30 21 1D 5B CD 7D 5E 3A F2 : 6D
FC38 5E 32 0E 5B C9 2A 03 5B : 4A
FC40 22 E8 5E DD 21 00 80 ED : D3
FC48 4B 07 5B C5 CD 10 5E CD : 7A
FC50 CE 5C 2A E8 5E 23 22 E8 : C7
FC58 5E C1 0B 78 B1 20 EC CD : 2C
FC60 99 5C 18 CC 3A 10 5B 1E : 9C
FC68 04 0F D8 21 2B 5B CD 7D : DC
FC70 5E 1E 03 C9 3A E6 5E 32 : F8
FC78 E5 5E 3A 10 5B 0F D0 21 : E8
---------------------------------
SUM: 7A 3E 9C B6 C9 CE F8 F4 2E17

FC80 2B 5B CD 7D 5E 3A E6 5E : AC
FC88 21 85 5E 87 5F 16 00 19 : 19
FC90 7E 23 32 E5 5E 7E DF 06 : 79
FC98 C9 21 25 5B CD 7D 5E 3A : 4C
FCA0 10 5B 0F 30 08 21 F7 5E : 28
FCA8 CD 7D 5E 18 0B 06 02 21 : F4
FCB0 F6 5E 7E 2B DF 06 10 FA : EC
FCB8 21 00 80 ED 4B F3 5E 7E : A8
FCC0 DF 06 23 0B 78 B1 20 F7 : 53
FCC8 21 21 5B C3 7D 5E 3A 0F : 84
FCD0 5B B7 08 3A 0B 5B CB 3F : C4
FCD8 CB 3F CB 3F 4F 3A E7 5E : E2
FCE0 47 50 59 3A E5 5E FD 21 : 8B
FCE8 10 5B 21 FF 5E DD E5 C5 : 70
FCF0 E5 F5 D5 08 28 06 08 CD : BA
FCF8 46 5D 18 04 08 CD 63 5D : 54
---------------------------------
SUM: 2F 74 A5 30 E7 1D E3 61 7C73

FD00 16 00 3A 0C 5B 4F 3A 0C : 4C
FD08 5B 47 7E 23 0F FD CB 00 : 1A
FD10 4E 28 06 DD CB 00 1E 18 : 5A
FD18 04 DD CB 00 16 10 ED DD : 9C
FD20 19 0D 20 E2 D1 F1 E1 01 : CC
FD28 03 00 09 C1 DD E1 10 BD : 58
FD30 DD 23 42 0D 20 B7 3A 0C : 6C
FD38 5B 3D 16 00 6A 62 CD 44 : 8B
FD40 01 EB DD 19 08 C9 CD B2 : 32
FD48 5D 47 3A 0C 5B FE 02 78 : BD
FD50 28 03 87 18 2B 3A E5 5E : 72
FD58 B7 78 20 1D FE 07 28 19 : B2
FD60 3C 18 16 CD 8C 5D 4F 3A : A9
FD68 0C 5B FE 02 20 08 CB 19 : 73
FD70 79 DE 00 87 18 03 79 18 : 8A
FD78 07 EE 07 21 95 5E 18 06 : 2E
---------------------------------
SUM: 1C A5 E3 8D 68 15 8F 21 0A09

FD80 EE 0F 87 21 A5 5E 87 4F : 7E
FD88 06 00 09 C9 D5 06 03 7E : 34
FD90 F5 23 10 FB 21 00 00 11 : 55
FD98 9A 00 F1 CD 44 01 11 2E : DC
FDA0 01 F1 CD 44 01 11 38 00 : 4D
FDA8 F1 CD 44 01 AF CB 25 8C : 2E
FDB0 D1 C9 FD CB 00 46 20 08 : D0
FDB8 D5 CD D2 5D D1 EE 07 C9 : 60
FDC0 B7 20 08 06 03 AF B6 23 : 70
FDC8 10 FC C9 06 00 3D 4F 09 : 70
FDD0 7E C9 4E 23 56 23 5E 06 : 95
FDD8 03 AF CB 1A 17 CB 1B 17 : AB
FDE0 CB 19 17 21 8D 5E 85 6F : FB
FDE8 3E 00 8C 67 7E F5 10 E9 : 9D
FDF0 06 03 0E 00 51 59 F1 1F : D1
FDF8 CB 11 1F CB 12 1F CB 13 : D5
---------------------------------
SUM: 3D 47 2B BB 3E 1A EE 3C CD5F

FE00 10 F4 3A E5 5E 3D 20 01 : DF
FE08 7B 3D 20 02 7A C9 79 C9 : 5F
FE10 2A 0D 5B E5 2A E8 5E 22 : 09
FE18 C0 0C 2A EA 5E 22 C2 0C : 2E
FE20 3A EE 5E 47 21 FF 5E C5 : 10
FE28 E5 21 C0 0C DF 4E 30 05 : 34
FE30 11 FF 01 18 04 0E 01 DF : 1B
FE38 5F E1 06 07 78 A3 77 23 : 02
FE40 7A 4B CB 23 17 CB 23 17 : CF
FE48 77 23 79 1F 1F 1F A0 77 : 87
FE50 23 EB 2A 0D 5B 7C 85 67 : 08
FE58 32 F2 5E 22 0D 5B 38 04 : 48
FE60 21 C2 0C 34 EB C1 10 BF : 9E
FE68 2A C2 0C 22 F0 5E E1 22 : 6B
FE70 0D 5B C9 7E B7 28 05 23 : B6
FE78 DF 06 18 F7 EB 7E B7 C8 : DC
---------------------------------
SUM: 81 69 C9 64 F7 94 EC 89 3BC5

FE80 23 DF 06 18 F8 00 30 01 : 49
FE88 36 02 33 03 35 07 01 06 : B1
FE90 02 05 03 04 00 00 00 00 : 0E
FE98 01 00 01 01 01 01 02 01 : 08
FEA0 03 01 03 03 03 00 00 00 : 0D
FEA8 00 00 00 00 00 00 04 00 : 04
FEB0 00 00 06 00 00 00 06 02 : 0E
FEB8 00 00 06 06 00 00 06 0E : 20
FEC0 00 00 0E 0E 00 08 0E 0E : 40
FEC8 00 0E 0E 0E 00 0F 0E 0E : 55
FED0 00 0F 0F 0F 00 0F 0F 0F : 5A
FED8 01 0F 0F 0F 03 0F 0F 0F : 5E
FEE0 07 0F 0F 0F 0F 00 00 00 : 43
FEE8 00 00 00 00 00 00 00 00 : 00
---------------------------------
SUM: 67 22 95 72 43 3D 7D 52 7BEB
```

リスト4 ソースリスト(参考)

```
0000:                   0001: ;# `/画餅/印刷/hcopy.s` by motoha-j
0000:                   0002: ;#   Apr 28 1989 0:55 a.m.
0000:                   0003: ;#
0000:                   0004: ;MZ-2500 system routine
0000:                   0005: _mulux  EQU     0144H           ;HL -- DE * A;
0000:                   0006: _divu   EQU     0153H           ;HL /- DE; DE - HL % DE;
0000:                   0007: GRPWK   EQU     0CC0H           ;グラフィック用ワーク
0000:                   0008: ;IOCS(svc) call
0000:                   0009: #LPOUT  EQU     06H             ;lpout
0000:                   0010: #POINT  EQU     4EH             ;A - point((DE+0),(DE+2));
0000:                   0011: #COCNV  EQU     5FH             ;DE(GRBcode) - A(1byte code);
0000:                   0012:         ;
0000:                   0013:         ORG     05B00H,0FB00H   ;ORG 実際の実行アドレス,格納
アドレス
5B00:                   0014:
5B00: C3 31 5B          0015:         JP      HCOPY
5B03:                   0016: ;画餅と連絡をとるワーク
5B03:                   0017: ;%%%%%%%.work
5B03:                   0018: lpx0:   DS      2  ;x
5B05:                   0019: lpy0:   DS      2  ;y
5B07:                   0020: lplx:   DS      2  ;lx
5B09:                   0021: lply:   DS      2  ;ly
5B0B: 18                0022: lppin:  DB      24 ;pin: 8/16/24
5B0C:                   0023: lpdot:  DS      1  ;2:small/4:large
5B0D:                   0024: lpsst:  DS      2  ;L - ST(x*256)-(1.x-横 / 縦); H- L-1;
5B0F:                   0025: lpmd:   DS      1  ;mode-  0:mono / 1:color
5B10: 00                0026: printe: DB      0  ;bit0/ 0:MZ-1P17etc. 1:PC-PR406
5B11:                   0027: ;                  ;bit1/ 一番上が  0:最上位bit   1:最下位bit
5B11:                   0028: ;%%%%%%%.data   ;MZ-1P17の設定  ;PC-PR406の設定
5B11: 0A 00 00 00       0029: LCOPYM: DB      0AH,0,0,0,0,0   ;1BH,"D",0,0,0,0  ;copy modeの設
定
5B15: 00 00
5B17: 1B 25 36 10       0030: LLFDAT: DB      1BH,"%6",16,0,0 ;1BH,"T","18",0,0 ;LFの幅の設定
5B1B: 00 00
5B1D: 0A 00 00 00       0031: LLF:    DB      0AH,0,0,0       ;LF code
5B21: 0D 00 00 00       0032: LCR:    DB      0DH,0,0,0       ;CR code
5B25: 1B 25 31 00       0033: LBITIM: DB      1BH,"%1",0,0,0  ;1BH,"J",0,0,0,0 ;bit imageの設
定
5B29: 00 00
5B2B: 1B 19 00 00       0034: LCOLDT: DB      1BH,19H,0,0,0,0 ;1BH,"C",0,0,0,0 ;color mode
5B2F: 00 00
5B31:                   0035: ;HCOPY ####
5B31:                   0036: ;##########
5B31: CD 38 5B          0037: HCOPY:  CALL    LINIT
5B34: CD E1 5B          0038:         CALL    LPRTY
5B37: C9                0039:         RET
5B38:                   0040: ;initialize /out:B(Y loop_counter)
5B38:                   0041: ;----------------
5B38: 3A 0B 5B          0042: LINIT:  LD      A,(lppin)       ;ヘッドのピン数
5B3B: CB 3F             0043:         SRL     A               ;
5B3D: 5F                0044:         LD      E,A             ;
5B3E: 3A 0C 5B          0045:         LD      A,(lpdot)       ;小型-2,大型-4
5B41: FE 02             0046:         CP      2               ;だから結局
5B43: 20 02             0047:         JR      NZ,LINIT0       ;
5B45: CB 3B             0048:         SRL     E               ;
5B47: 7B                0049: LINIT0: LD      A,E             ;((lppin/8)バイト)*lpdot
5B48: ED 5B 07 5B       0050:         LD      DE,(lplx)       ;横のドット数
5B4C: 21 00 00          0051:         LD      HL,0            ;
5B4F: CD 44 01          0052:         CALL    _mulux          ;(lplx*lpdot*lppin/8) bytes
5B52: 22 F0 5E          0053:         LD      (LPTLEN),HL     ;-1回のヘッドの移動での全
バイト数
5B55:                   0054:         ;
5B55: 2A 07 5B          0055:         LD      HL,(lplx)
5B58: 29                0056:         ADD     HL,HL
5B59: 3A 0C 5B          0057:         LD      A,(lpdot)
5B5C: FE 02             0058:         CP      2
5B5E: 28 01             0059:         JR      Z,LINIT1
5B60: 29                0060:         ADD     HL,HL           ;lplx*lpdot
5B61: 22 F2 5E          0061: LINIT1: LD      (LPTLX),HL      ;プリンタの横ドット(ピン)
数
5B64: 3A 10 5B          0062:         LD      A,(printe)      ;プリンタコード
5B67: CB 47             0063:         BIT     0,A             ;
5B69: 28 19             0064:         JR      Z,LINIT4        ;
5B6B:                   0065:         ;///////////////////////// PC-PR406
5B6B: 06 04             0066:         LD      B,4             ;プリンタの横ドット(ピン)
数
5B6D: 11 0A 00          0067: LINIT2: LD      DE,10           ;を文字型に変換している
5B70: CD 53 01          0068:         CALL    _divu
5B73: 7B                0069:         LD      A,E             ;DE - HL % DE;
5B74: C6 30             0070:         ADD     A,"0"
5B76: F5                0071:         PUSH    AF
5B77: 10 F4             0072:         DJNZ    LINIT2
5B79: 21 F4 5E          0073:         LD      HL,LPTWK
```

```
5B7C: 06 04         0074:         LD     B,4
5B7E: F1            0075: LINIT3: POP    AF
5B7F: 77            0076:         LD     (HL),A
5B80: 23            0077:         INC    HL
5B81: 10 FB         0078:         DJNZ   LINIT3
5B83: 70            0079:         LD     (HL),B  ;-0     ;end code
5B84:               0080:         ;//////////////////////////////
5B84:               0081: LINIT4: ;initialize printer
5B84: 21 11 5B      0082:         LD     HL,LCOPYM
5B87: CD 7A 5E      0083:         CALL   LPTCOM          ;copy-mode
5B8A: 21 17 5B      0084:         LD     HL,LLFDAT
5B8D: CD 7A 5E      0085:         CALL   LPTCOM          ;LF の幅を設定
5B90: 2A 03 5B      0086:         LD     HL,(lpx0)
5B93: 22 E5 5E      0087:         LD     (lpx),HL        ;x
5B96: 2A 05 5B      0088:         LD     HL,(lpy0)
5B99: 22 E7 5E      0089:         LD     (lpy),HL        ;y
5B9C: AF            0090:         XOR    A
5B9D: 32 E3 5E      0091:         LD     (lpcnt),A       ;counter = 0;
5BA0: 2A 0D 5B      0092:         LD     HL,(lpsst)      ;ST=0.x*256
5BA3: 65            0093:         LD     H,L
5BA4: 25            0094:         DEC    H
5BA5: 22 0D 5B      0095:         LD     (lpsst),HL      ;S=ST-1
5BA8:               0096:         ;
5BA8: 3A 0D 5B      0097:         LD     A,(lpsst)       ;ST
5BAB: 5F            0098:         LD     E,A
5BAC: 16 00         0099:         LD     D,0
5BAE: 3A 09 5B      0100:         LD     A,(lply)        ;ly
5BB1: 67            0101:         LD     H,A             ;画面との比の補正
5BB2: 6A            0102:         LD     L,D             ;縦の比*長さ
5BB3: CD 44 01      0103:         CALL   _mulux          ;H=(ly+ly*0.x)=ly*(1.x)
5BB6: 3A 0C 5B      0104:         LD     A,(lpdot)
5BB9: FE 02         0105:         CP     2               ;lppin/lpdot
5BBB: 3A 0B 5B      0106:         LD     A,(lppin)
5BBE: 0E 08         0107:         LD     C,8
5BC0: 28 04         0108:         JR     Z,LINIT5
5BC2: CB 3F         0109:         SRL    A
5BC4: CB 39         0110:         SRL    C
5BC6: CB 3F         0111: LINIT5: SRL    A
5BC8: CB 39         0112:         SRL    C               ;ヘッド1列分の画面上のドッ
ト数
5BCA: 32 EB 5E      0113:         LD     (lpdy),A        ;lppin/lpdot
5BCD: 5F            0114:         LD     E,A
5BCE: 79            0115:         LD     A,C
5BCF: 32 E4 5E      0116:         LD     (lp8dv),A       ;8/lpdot
5BD2:               0117:         ;
5BD2: 6C            0118:         LD     L,H
5BD3: 26 00         0119:         LD     H,0
5BD5: 54            0120:         LD     D,H
5BD6: CD 53 01      0121:         CALL   _divu           ;ly*1.x/(pin/dot)
5BD9: 7B            0122:         LD     A,E             ;補正した長さ/ヘッド1列分
の画面上のドット数
5BDA: B7            0123:         OR     A               ; =ループ回数
5BDB: 28 01         0124:         JR     Z,LINIT6
5BDD: 2C            0125:         INC    L
5BDE: 45            0126: LINIT6: LD     B,L             ;Y loop counter
5BDF: 04            0127:         INC    B
5BE0: C9            0128:         RET
5BE1:               0129: ;メイン----------
5BE1:               0130: ;----------------
5BE1: C5            0131: LPRTY:  PUSH   BC              ;loop counter
5BE2:               0132:         ;
5BE2: 3A 0F 5B      0133:         LD     A,(lpmd)        ;白黒/天然?
5BE5: B7            0134:         OR     A
5BE6: 28 05         0135:         JR     Z,LPRTY1
5BE8: CD FA 5B      0136:         CALL   LPTXc           ;color
5BEB: 18 03         0137:         JR     LPRTY2
5BED: CD 3D 5C      0138: LPRTY1: CALL   LPTXm           ;mono
5BF0: 2A ED 5E      0139: LPRTY2: LD     HL,(lpy1)
5BF3: 22 E7 5E      0140:         LD     (lpy),HL        ;y 座標
5BF6:               0141:         ;
5BF6: C1            0142:         POP    BC
5BF7: 10 E8         0143:         DJNZ   LPRTY
5BF9: C9            0144:         RET
5BFA:               0145: ;天然モードの1回の移動
5BFA:               0146: ;---------------------
5BFA: CD 64 5C      0147: LPTXc:  CALL   LPXcs           ;天然モード初期化
5BFD: D5            0148: LPTXc1: PUSH   DE              ;E=カラーリボンの色数
5BFE:               0149:         ;setting color LPT
5BFE: 3E 04         0150:         LD     A,4
5C00: 93            0151:         SUB    E
5C01: 32 E3 5E      0152:         LD     (lpcnt),A       ;カラー番号
5C04: CD 74 5C      0153:         CALL   LPXcCL
5C07:               0154:         ;
5C07: 2A 03 5B      0155:         LD     HL,(lpx0)
5C0A: 22 E5 5E      0156:         LD     (lpx),HL        ;x 座標初期化
5C0D: DD 21 00 80   0157:         LD     IX,LPTBF        ;プリント用バッファポインタ
5C11: ED 4B 07 5B   0158:         LD     BC,(lplx)       ;lx
5C15: C5            0159: LPTXc2: PUSH   BC
5C16:               0160:         ;
5C16: CD 0D 5E      0161:         CALL   LGRDT           ;GRAM->データ形式の変換
5C19: CD CB 5C      0162:         CALL   LCNG            ;画面輝度->printer輝度
5C1C: 2A E5 5E      0163:         LD     HL,(lpx)
5C1F: 23            0164:         INC    HL
5C20: 22 E5 5E      0165:         LD     (lpx),HL        ;x++
5C23:               0166:         ;
5C23: C1            0167:         POP    BC
5C24: 0B            0168:         DEC    BC              ;lx--
5C25: 78            0169:         LD     A,B
5C26: B1            0170:         OR     C
5C27: 20 EC         0171:         JR     NZ,LPTXc2
5C29: CD 99 5C      0172:         CALL   LPTWRT          ;一挙に印刷!
5C2C:               0173:         ;
5C2C: D1            0174:         POP    DE
5C2D: 1D            0175:         DEC    E               ;インクリボン counter--
5C2E: 20 CD         0176:         JR     NZ,LPTXc1
5C30: 21 1D 5B      0177: LPTXc3: LD     HL,LLF          ;ラインフィード指令
5C33: CD 7A 5E      0178:         CALL   LPTCOM          ;LF
5C36: 3A EF 5E      0179:         LD     A,(lps1)
5C39: 32 0E 5B      0180:         LD     (lpsst+1),A     ;縦比率用S,ST
5C3C: C9            0181:         RET
5C3D:               0182: ;白黒モードの1回の移動
5C3D:               0183: ;---------------------
5C3D: 2A 03 5B      0184: LPTXm:  LD     HL,(lpx0)
5C40: 22 E5 5E      0185:         LD     (lpx),HL        ;x 座標初期化
5C43: DD 21 00 80   0186:         LD     IX,LPTBF        ;プリント用バッファポインタ
5C47: ED 4B 07 5B   0187:         LD     BC,(lplx)       ;lx
5C4B: C5            0188: LPTXm1: PUSH   BC
5C4C:               0189:         ;
5C4C: CD 0D 5E      0190:         CALL   LGRDT           ;GRAM->データ形式の変換
5C4F: CD CB 5C      0191:         CALL   LCNG            ;画面輝度->printer輝度
5C52: 2A E5 5E      0192:         LD     HL,(lpx)        ;x++
5C55: 23            0193:         INC    HL
5C56: 22 E5 5E      0194:         LD     (lpx),HL
5C59:               0195:         ;
5C59: C1            0196:         POP    BC              ;lx--
5C5A: 0B            0197:         DEC    BC
5C5B: 78            0198:         LD     A,B
5C5C: B1            0199:         OR     C
5C5D: 20 EC         0200:         JR     NZ,LPTXm1
5C5F: CD 99 5C      0201:         CALL   LPTWRT          ;一挙に印字だぁぁぁ!!
5C62: 18 CC         0202:         JR     LPTXc3
5C64:               0203: ;天然モード初期化
5C64:               0204: ;----------------
5C64: 3A 10 5B      0205: LPXcs:  LD     A,(printe)
5C67: 1E 04         0206:         LD     E,4             ;リボンは4色
5C69: 0F            0207:         RRCA
5C6A: D8            0208:         RET    C               ;PC-PR406なら RET
5C6B: 21 2B 5B      0209:         LD     HL,LCOLDT       ;/// MZ-1P17
5C6E: CD 7A 5E      0210:         CALL   LPTCOM          ;    カラー印字指令
5C71: 1E 03         0211:         LD     E,3             ;    リボンは3色 ///
5C73: C9            0212:         RET
5C74:               0213: ;color data =>LPT------
5C74:               0214: ;----------------------
5C74: 3A E3 5E      0215: LPXcCL: LD     A,(lpcnt)       ;カラー番号
5C77: 32 E2 5E      0216:         LD     (lpcol),A       ;カラー
5C7A: 3A 10 5B      0217:         LD     A,(printe)
5C7D: 0F            0218:         RRCA
5C7E: D0            0219:         RET    NC              ;/// MZ-1P17なら RET ///
5C7F:               0220:         ;----------------      ;/// PC-PR406
5C7F: 21 2B 5B      0221:         LD     HL,LCOLDT
5C82: CD 7A 5E      0222:         CALL   LPTCOM          ;カラーリボンの選択
5C85: 3A E3 5E      0223:         LD     A,(lpcnt)       ;カラー番号
5C88: 21 82 5E      0224:         LD     HL,LCOLOR       ;表から
5C8B: 87            0225:         ADD    A,A             ;実際の
5C8C: 5F            0226:         LD     E,A             ;カラー番号を
5C8D: 16 00         0227:         LD     D,0             ;得る
5C8F: 19            0228:         ADD    HL,DE
5C90: 7E            0229:         LD     A,(HL)
5C91: 23            0230:         INC    HL
5C92: 32 E2 5E      0231:         LD     (lpcol),A       ;実際のカラー番号
5C95: 7E            0232:         LD     A,(HL)          ;HL=color setting data
5C96: DF 06         0233:         .SVC   #LPOUT          ;指令を送る
5C98: C9            0234:         RET
5C99:               0235: ;Mach Go!!Go!!Go!!! --
5C99:               0236: ;---------------------
5C99:               0237: LPTWRT: ;bit image command
5C99: 21 25 5B      0238:         LD     HL,LBITIM       ;ビットイメージモード
5C9C: CD 7A 5E      0239:         CALL   LPTCOM          ;設定だぁぁぁ
5C9F: 3A 10 5B      0240:         LD     A,(printe)
5CA2: 0F            0241:         RRCA
5CA3: 30 05         0242:         JR     NC,LPTWR1
5CA5: CD 7A 5E      0243:         CALL   LPTCOM          ;//// PC-PR406
5CA8: 18 0B         0244:         JR     LPTWR3          ;ビットイメージ長さ ////
5CAA: 06 02         0245: LPTWR1: LD     B,2
5CAC: 21 F3 5E      0246:         LD     HL,LPTLX+1
5CAF: 7E            0247: LPTWR2: LD     A,(HL)          ;//// MZ-1P17
5CB0: 2B            0248:         DEC    HL              ;ビットイメージ長さ ////
5CB1: DF 06         0249:         .SVC   #LPOUT
5CB3: 10 FA         0250:         DJNZ   LPTWR2
5CB5:               0251:         ;print out!            ;プリンタへだだーっと
5CB5: 21 00 80      0252: LPTWR3: LD     HL,LPTBF        ;データを送る
5CB8: ED 4B F0 5E   0253:         LD     BC,(LPTLEN)     ;BC =プリントデータの長さ
5CBC: 7E            0254: LPTWR4: LD     A,(HL)
5CBD: DF 06         0255:         .SVC   #LPOUT
5CBF: 23            0256:         INC    HL
5CC0: 0B            0257:         DEC    BC
5CC1: 78            0258:         LD     A,B
5CC2: B1            0259:         OR     C
5CC3: 20 F7         0260:         JR     NZ,LPTWR4
5CC5:               0261:         ;CR
5CC5: 21 21 5B      0262:         LD     HL,LCR          ;印字(CR)指令を送る
5CC8: C3 7A 5E      0263:         JP     LPTCOM
5CCB:               0264: ;GRBデータ => プリンタ用データに変換
5CCB:               0265: ; in: IX(print buffer)
5CCB:               0266: ;--------------------
5CCB: 3A 0F 5B      0267: LCNG:   LD     A,(lpmd)        ;*2 or *4
5CCE: B7            0268:         OR     A               ;0:白黒/1:天然
5CCF: 08            0269:         EX     AF,AF'  ;{af
5CD0: 3A 0B 5B      0270:         LD     A,(lppin)
5CD3: CB 3F         0271:         SRL    A
5CD5: CB 3F         0272:         SRL    A
5CD7: CB 3F         0273:         SRL    A               ;lppin/8 =縦のバイト数
5CD9: 4F            0274:         LD     C,A
5CDA: 3A E4 5E      0275:         LD     A,(lp8dv)       ;8/lpdot =1バイト毎の画面上の
ドット数
5CDD: 47            0276:         LD     B,A
5CDE: 50 59         0277:         LD     DE,BC
5CE0: 3A E2 5E      0278:         LD     A,(lpcol)       ;BYMS(or YMS) color
5CE3: FD 21 10 5B   0279:         LD     IY,printe       ;プリンタコード
5CE7: 21 FC 5E      0280:         LD     HL,LGRPDT       ;GRB毎のデータエリア
5CEA: DD E5         0281: LCNG1:  PUSH   IX              ;プリント用バッファポインタ
5CEC: C5            0282:         PUSH   BC              ;B=8/lpdot
5CED: E5            0283:         PUSH   HL              ;HL=GRB毎のデータエリア
5CEE: F5            0284:         PUSH   AF              ; A=color
5CEF: D5            0285:         PUSH   DE              ;D=8/lpdot,E=lppin/8
5CF0:               0286:         ;
5CF0: 08            0287:         EX     AF,AF'  ;af}
5CF1: 28 06         0288:         JR     Z,LCNG2         ;白黒なら
5CF3: 08            0289:         EX     AF,AF'  ;{af
5CF4: CD 43 5D      0290:         CALL   LCNGc           ;天然なら
5CF7: 18 04         0291:         JR     LCNG3
5CF9: 08            0292: LCNG2:  EX     AF,AF'  ;{af
5CFA: CD 60 5D      0293:         CALL   LCNGm           ;白黒なら
5CFD:               0294: LCNG3:  ;                      ;HL(font address pointer)
5CFD: 16 00         0295:         LD     D,0             ;DE=pin/8
5CFF: 3A 0C 5B      0296:         LD     A,(lpdot)
5D02: 4F            0297:         LD     C,A             ;カウンタ1
5D03: 3A 0C 5B      0298: LCNG4:  LD     A,(lpdot)
5D06: 47            0299:         LD     B,A             ;カウンタ2
5D07: 7E            0300:         LD     A,(HL)          ;get font
5D08: 23            0301:         INC    HL
5D09: 0F            0302: LCNG5:  RRCA                   ;Font data 下位bitから取り出
すぅ
5D0A: FD CB 00 4E   0303:         BIT    1,(IY+0)        ;方向モード
5D0E: 28 06         0304:         JR     Z,LCNG6
5D10: DD CB 00 1E   0305:         RR     (IX+0)          ;上から順に最下位bit,...,最上
位bit
5D14: 18 04         0306:         JR     LCNG7
5D16: DD CB 00 16   0307: LCNG6:  RL     (IX+0)          ;上から順に最上位bit,...,最下
位bit
5D1A: 10 ED         0308: LCNG7:  DJNZ   LCNG5
5D1C: DD 19         0309:         ADD    IX,DE           ;next 縦 line
5D1E: 0D            0310:         DEC    C
5D1F: 20 E2         0311:         JR     NZ,LCNG4
5D21:               0312:         ;
5D21: D1            0313:         POP    DE              ;ぐぬぬぬぬぬぬぬ
5D22: F1            0314:         POP    AF              ;         pop
5D23: E1            0315:         POP    HL
5D24: 01 03 00      0316:         LD     BC,3
5D27: 09            0317:         ADD    HL,BC           ;next GRBデータエリア
5D28: C1            0318:         POP    BC
5D29: DD E1         0319:         POP    IX
5D2B: 10 BD         0320:         DJNZ   LCNG1
5D2D: DD 23         0321:         INC    IX              ;next 横 line
5D2F: 42            0322:         LD     B,D
5D30: 0D            0323:         DEC    C
5D31: 20 B7         0324:         JR     NZ,LCNG1
5D33:               0325:         ;
5D33: 3A 0C 5B      0326:         LD     A,(lpdot)       ;2:small / 4:large
5D36: 3D            0327:         DEC    A
5D37: 16 00         0328:         LD     D,0             ;DE=lppin/8 バイト
5D39: 6A            0329:         LD     L,D
5D3A: 62            0330:         LD     H,D
5D3B: CD 44 01      0331:         CALL   _mulux          ;HL:=(lppin/8)*(lpdot-1)
5D3E: EB            0332:         EX     DE,HL           ;次のxのための
5D3F: DD 19         0333:         ADD    IX,DE           ;プリントバッファ位置へポイ
ンタを移動
5D41: 08            0334:         EX     AF,AF'  ;af}
5D42: C9            0335:         RET
5D43:               0336: ;get ドットフォント for 天然色mode
5D43:               0337: ; in:HL(RGB),A(color)
5D43:               0338: ;out:HL(font adr.)
5D43:               0339: ;----------------------
5D43: CD AF 5D      0340: LCNGc:  CALL   LCNGcs          ;輝度計算(for COLOR)
5D46:               0341:         ;
5D46: 47            0342:         LD     B,A             ;B=輝度
5D47: 3A 0C 5B      0343:         LD     A,(lpdot)
5D4A: FE 02         0344:         CP     2
5D4C: 78            0345:         LD     A,B             ;A=輝度
5D4D: 28 03         0346:         JR     Z,LCNGc5
```


▶F1ポルトガルGP，あの事故は誰もが腹を立てたはずだ。私はセナのファンというわけではないが，それにしてもあのマンセルの行為は許せない。なんとも後味のわるいレースだった。ひいきのルノーもニューマシンが不調だったし……でも中嶋がよくがんばってくれた。よかった。日本GPに大きな期待！ 山下 智也 (19) 大阪府

```
5D4F:                0347:          ;large                    ;大型サイズ
5D4F: 87             0348:          ADD     A,A               ;戻り値を16階調にするため2
倍する
5D50: 18 2B          0349:          JR      LCNG1
5D52:                0350: LCNGc5:  ;small
5D52: 3A E2 5E       0351:          LD      A,(lpcol)         ;カラー
5D55: B7             0352:          OR      A                 ;
5D56: 78             0353:          LD      A,B               ;A-輝度
5D57: 20 1D          0354:          JR      NZ,LCNGs
5D59: FE 07          0355:          CP      7                 ;輝度のいい加減な補正
5D5B: 28 19          0356:          JR      Z,LCNGs
5D5D: 3C             0357:          INC     A
5D5E: 18 16          0358:          JR      LCNGs
5D60:                0359: ;get ドットフォント for 白黒モード
5D60:                0360: ; in:HL(RGB) out:HL(font adr.)
5D60:                0361: ;----------------------
5D60: CD 89 5D       0362: LCNGm:   CALL    LBRIG             ;計算
5D63: 4F             0363:          LD      C,A               ;輝度
5D64: 3A 0C 5B       0364:          LD      A,(lpdot)
5D67: FE 02          0365:          CP      2
5D69: 20 08          0366:          JR      NZ,LCNGm1
5D6B:                0367:          ;small
5D6B: CB 19          0368:          RR      C                 ;A-(A/2-A%2)*2
5D6D: 79             0369:          LD      A,C
5D6E: DE 00          0370:          SBC     A,0
5D70: 87             0371:          ADD     A,A
5D71: 18 03          0372:          JR      LCNGs
5D73:                0373: LCNGm1:  ;large
5D73: 79             0374:          LD      A,C               ;輝度
5D74: 18 07          0375:          JR      LCNG1
5D76:                0376: ;sub --
5D76:                0377: ;------
5D76: EE 07          0378: LCNGs:   XOR     07H               ;輝度データを反転し濃度へ
5D78: 21 92 5E       0379:          LD      HL,LFONTs         ;フォント小
5D7B: 18 06          0380:          JR      LCNG1s
5D7D:                0381:          ;
5D7D: EE 0F          0382: LCNG1:   XOR     0FH               ;輝度データを反転し濃度へ
5D7F: 87             0383:          ADD     A,A
5D80: 21 A2 5E       0384:          LD      HL,LFONT1         ;フォント大
5D83: 87             0385: LCNG1s:  ADD     A,A               ;輝度 * lpdot + フォントアド
レス
5D84: 4F             0386:          LD      C,A
5D85: 06 00          0387:          LD      B,0
5D87: 09             0388:          ADD     HL,BC
5D88: C9             0389:          RET
5D89:                0390: ;白黒モードのプリントデータを得る
5D89:                0391: ; in:HL(RGB) out:HL(font adr.)
5D89:                0392: ;----------------------
5D89: D5             0393: LBRIG:   PUSH    DE                ;白黒モード
5D8A:                0394:          ;                         ;階調計算
5D8A: 06 03          0395:          LD      B,3
5D8C: 7E             0396: LBRIG1:  LD      A,(HL)
5D8D: F5             0397:          PUSH    AF
5D8E: 23             0398:          INC     HL
5D8F: 10 FB          0399:          DJNZ    LBRIG1
5D91:                0400:          ;
5D91: 21 00 00       0401:          LD      HL,0
5D94: 11 9A 00       0402:          LD      DE,009AH          ;0.30*512
5D97: F1             0403:          POP     AF                ;R * 0.3
5D98: CD 44 01       0404:          CALL    _mulux            ;HL+DE*A
5D9B: 11 2E 01       0405:          LD      DE,012EH          ;0.59*512
5D9E: F1             0406:          POP     AF                ;G * 0.59
5D9F: CD 44 01       0407:          CALL    _mulux            ;HL+DE*A
5DA2: 11 38 00       0408:          LD      DE,0038H          ;0.11*512
5DA5: F1             0409:          POP     AF                ;B * 0.11
5DA6: CD 44 01       0410:          CALL    _mulux            ;HL+DE*A -階調
5DA9: AF             0411:          XOR     A
5DAA: CB 25          0412:          SLA     L
5DAC: 8C             0413:          ADC     A,H               ;A - AL*2;
5DAD:                0414:          ;
5DAD: D1             0415:          POP     DE
5DAE: C9             0416:          RET
5DAF:                0417: ;color change GRB->YMSK(or YMB)
5DAF:                0418: ;--------------------------------
5DAF: FD CB 00 46    0419: LCNGcs:  BIT     0,(IY+0)
5DB3: 20 08          0420:          JR      NZ,LCNGpc         ;PC-PR406用設定
5DB5: D5             0421:          PUSH    DE
5DB6: CD CF 5D       0422:          CALL    LCNGmz            ;MZ-1P17用設定
5DB9: D1             0423:          POP     DE
5DBA: EE 07          0424:          XOR     07H
5DBC: C9             0425:          RET
5DBD:                0426: ;------PC-PR406                      ;///日電
5DBD: B7             0427: LCNGpc:  OR      A                 ;カラー番号
5DBE: 20 08          0428:          JR      NZ,LCNGp2
5DC0:                0429:          ;black                    ;カラー番号-0なら黒リボン
5DC0: 06 03          0430:          LD      B,3
5DC2: AF             0431:          XOR     A
5DC3: B6             0432: LCNGp1:  OR      (HL)              ;GRB dataのORをとった後の0
のところが
5DC4: 23             0433:          INC     HL                ;黒でありまする.
5DC5: 10 FC          0434:          DJNZ    LCNGp1
5DC7: C9             0435:          RET
5DC8: 06 00          0436: LCNGp2:  LD      B,0               ;カラー番号-1:B
5DCA: 3D             0437:          DEC     A                 ;          2:R
5DCB: 4F             0438:          LD      C,A               ;          3:G
5DCC: 09             0439:          ADD     HL,BC
5DCD: 7E             0440:          LD      A,(HL)            ;///
5DCE: C9             0441:          RET
5DCF:                0442: ;------MZ-1P17                      ;/// 1P17様の設定
5DCF: 4E             0443: LCNGmz:  LD      C,(HL)   ;B
5DD0: 23             0444:          INC     HL
5DD1: 56             0445:          LD      D,(HL)   ;G
5DD2: 23             0446:          INC     HL
5DD3: 5E             0447:          LD      E,(HL)   ;R
5DD4: 06 03          0448:          LD      B,3
5DD6: AF             0449: LCNGz1:  XOR     A                 ;階調4,2,1とビット毎にまとめ
る
5DD7: CB 1A          0450:          RR      D        ;G
5DD9: 17             0451:          RLA
5DDA: CB 1B          0452:          RR      E        ;R
5DDC: 17             0453:          RLA
5DDD: CB 19          0454:          RR      C        ;B
5DDF: 17             0455:          RLA                       ;A-GRB順のビット構成
5DE0: 21 8A 5E       0456:          LD      HL,MZCOL          ;表のアドレス
5DE3: 85             0457:          ADD     A,L
5DE4: 6F             0458:          LD      L,A
5DE5: 3E 00          0459:          LD      A,0
5DE7: 8C             0460:          ADC     A,H
5DE8: 67             0461:          LD      H,A
5DE9: 7E             0462:          LD      A,(HL)            ;A-YMB順のビット構成
5DEA: F5             0463:          PUSH    AF                ;1,2,4 PUSH
5DEB: 10 E9          0464:          DJNZ    LCNGz1
5DED:                0465:          ;
5DED: 06 03          0466:          LD      B,3
5DEF: 0E 00          0467:          LD      C,0
5DF1: 51             0468:          LD      D,C               ;C- D- E- 0;
5DF2: 59             0469:          LD      E,C
5DF3: F1             0470: LCNGz2:  POP     AF                ;4,2,1 POP
5DF4: 1F             0471:          RRA
5DF5: CB 11          0472:          RL      C        ;B
5DF7: 1F             0473:          RRA
5DF8: CB 12          0474:          RL      D        ;M
5DFA: 1F             0475:          RRA
5DFB: CB 13          0476:          RL      E        ;Y
5DFD: 10 F4          0477:          DJNZ    LCNGz2
5DFF:                0478:          ;
5DFF: 3A E2 5E       0479:          LD      A,(lpcol)         ;カラー番号
5E02: 3D             0480:          DEC     A
5E03: 20 01          0481:          JR      NZ,LCNGz3
5E05: 7B             0482:          LD      A,E      ;Y-1
5E06: 3D             0483: LCNGz3:  DEC     A
5E07: 20 02          0484:          JR      NZ,LCNGz4
5E09: 7A             0485:          LD      A,D      ;M-2
5E0A: C9             0486:          RET
5E0B: 79             0487: LCNGz4:  LD      A,C      ;B-3
5E0C: C9             0488:          RET
5E0D:                0489: ;graphic->GRBdata
5E0D:                0490: ;-----------------
5E0D: 2A 0D 5B       0491: LGRDT:   LD      HL,(lpsst)
5E10: E5             0492:          PUSH    HL                ;save S,ST
5E11:                0493:          ;
5E11: 2A E5 5E       0494:          LD      HL,(lpx)          ;座標(lpx,lpy)
5E14: 22 C0 0C       0495:          LD      (GRPWK),HL
5E17: 2A E7 5E       0496:          LD      HL,(lpy)
5E1A: 22 C2 0C       0497:          LD      (GRPWK+2),HL
5E1D: 3A EB 5E       0498:          LD      A,(lpdy)          ;1回のヘッド移動での
5E20:                0499:          ;                         ;画面上の縦のドット数が
5E20: 47             0500:          LD      B,A               ;カウンタとなるわけだこりゃ
5E21: 21 FC 5E       0501:          LD      HL,LGRPDT
5E24: C5             0502: LGRDT1:  PUSH    BC
5E25: E5             0503:          PUSH    HL
5E26: 21 C0 0C       0504:          LD      HL,GRPWK
5E29: DF 4E          0505:          .SVC    #POINT            ;A-point((HL+0),(HL+2))
5E2B: 30 05          0506:          JR      NC,LGRDT2         ;out of view ならカラーは
5E2D: 11 FF 01       0507:          LD      DE,1FFH           ;0777(-真っ白)とみなす
5E30: 18 04          0508:          JR      LGRDT3
5E32: 0E 01          0509: LGRDT2:  LD      C,1               ;A-color
5E34: DF 5F          0510:          .SVC    #COCNV            ;-> DE-GRB順に全9ビット
5E36:                0511: LGRDT3:  ;                              -g2g1g0r2r1r0b2b1b0
5E36: E1             0512:          POP     HL
5E37: 06 07          0513:          LD      B,07H
5E39: 78             0514:          LD      A,B
5E3A: A3             0515:          AND     E
5E3B: 77             0516:          LD      (HL),A            ;Bデータ設定
5E3C: 23             0517:          INC     HL
5E3D: 7A             0518:          LD      A,D
5E3E: 4B             0519:          LD      C,E
5E3F: CB 23          0520:          SLA     E
5E41: 17             0521:          RLA
5E42: CB 23          0522:          SLA     E
5E44: 17             0523:          RLA
5E45: 77             0524:          LD      (HL),A            ;Gデータ設定
5E46: 23             0525:          INC     HL
5E47: 79             0526:          LD      A,C
5E48: 1F             0527:          RRA
5E49: 1F             0528:          RRA
5E4A: 1F             0529:          RRA
5E4B: A0             0530:          AND     B
5E4C: 77             0531:          LD      (HL),A            ;Rデータ設定
5E4D: 23             0532:          INC     HL
5E4E: EB             0533:          EX      DE,HL
5E4F:                0534:          ;
5E4F: 2A 0D 5B       0535:          LD      HL,(lpsst)
5E52: 7C             0536:          LD      A,H               ;S-S+ST
5E53: 85             0537:          ADD     A,L
5E54: 67             0538:          LD      H,A
5E55: 32 EF 5E       0539:          LD      (lps1),A          ;save 次のS
5E58: 22 0D 5B       0540:          LD      (lpsst),HL        ;save 次のSST
5E5B: 38 04          0541:          JR      C,LGRDT4
5E5D: 21 C2 0C       0542:          LD      HL,GRPWK+2
5E60: 34             0543:          INC     (HL)              ;lpy++
5E61: EB             0544: LGRDT4:  EX      DE,HL
5E62: C1             0545:          POP     BC
5E63: 10 BF          0546:          DJNZ    LGRDT1
5E65:                0547:          ;
5E65: 2A C2 0C       0548:          LD      HL,(GRPWK+2)
5E68: 22 ED 5E       0549:          LD      (lpy1),HL         ;next y座標
5E6B: E1             0550:          POP     HL
5E6C: 22 0D 5B       0551:          LD      (lpsst),HL        ;元のSSTに戻す
5E6F: C9             0552:          RET
5E70:                0553: ;プリンタへデータを送る
5E70:                0554: ;---------------------
5E70:                0555: SETLPT:  ;in:HL(command),DE(parameter)
5E70: 7E             0556:          LD      A,(HL)
5E71: B7             0557:          OR      A                 ;A-0; end code
5E72: 28 05          0558:          JR      Z,SETLT1
5E74: 23             0559:          INC     HL
5E75: DF 06          0560:          .SVC    #LPOUT
5E77: 18 F7          0561:          JR      SETLPT
5E79: EB             0562: SETLT1:  EX      DE,HL
5E7A:                0563:          ;
5E7A:                0564: LPTCOM:  ;in:HL(data)
5E7A: 7E             0565:          LD      A,(HL)
5E7B: B7             0566:          OR      A                 ;A-0; end code
5E7C: C8             0567:          RET     Z
5E7D: 23             0568:          INC     HL
5E7E: DF 06          0569:          .SVC    #LPOUT
5E80: 18 F8          0570:          JR      LPTCOM
5E82:                0571: ;%%%%%%%.data
5E82: 00 30          0572: LCOLOR:  DB      0,"0"    ;k 黒    ;PC-PR406
5E84: 01 36          0573:          DB      1,"6"    ;y 黄
5E86: 02 33          0574:          DB      2,"3"    ;m マゼンダ
5E88: 03 35          0575:          DB      3,"5"    ;c シアン
5E8A: 07 01 06 02    0576: MZCOL:   DB      7,1,6,2,5,3,4,0 ;MZ-1P17 GRB->BMY変換表
5E8E: 05 03 04 00
5E92:                0577: LFONTs:  ;small mode
5E92: 00 00          0578:          DB      00H,00H           ;マトリクスデータ
5E94: 00 01          0579:          DB      00H,01H
5E96: 00 01          0580:          DB      00H,01H
5E98: 01 01          0581:          DB      01H,01H
5E9A: 01 02          0582:          DB      01H,02H
5E9C: 01 03          0583:          DB      01H,03H
5E9E: 01 03          0584:          DB      01H,03H
5EA0: 03 03          0585:          DB      03H,03H
5EA2:                0586: LFONT1:  ;large mode
5EA2: 00 00 00 00    0587:          DB      00H,00H,00H,00H ;0
5EA6: 00 00 00 00    0588:          DB      00H,00H,00H,00H
5EAA: 00 04 00 00    0589:          DB      00H,04H,00H,00H
5EAE: 00 06 00 00    0590:          DB      00H,06H,00H,00H
5EB2: 00 06 02 00    0591:          DB      00H,06H,02H,00H
5EB6: 00 06 06 00    0592:          DB      00H,06H,06H,00H
5EBA: 00 06 0E 00    0593:          DB      00H,06H,0EH,00H
5EBE: 00 0E 0E 00    0594:          DB      00H,0EH,0EH,00H
5EC2: 08 0E 0E 00    0595:          DB      08H,0EH,0EH,00H ;8
5EC6: 0E 0E 0E 00    0596:          DB      0EH,0EH,0EH,00H
5ECA: 0F 0E 0E 00    0597:          DB      0FH,0EH,0EH,00H
5ECE: 0F 0F 0F 00    0598:          DB      0FH,0FH,0FH,00H
5ED2: 0F 0F 0F 01    0599:          DB      0FH,0FH,0FH,01H
5ED6: 0F 0F 0F 03    0600:          DB      0FH,0FH,0FH,03H
5EDA: 0F 0F 0F 07    0601:          DB      0FH,0FH,0FH,07H
5EDE: 0F 0F 0F 0F    0602:          DB      0FH,0FH,0FH,0FH
5EE2:                0603: ;%%%%%%%.work
5EE2:                0604: lpcol:   DS      1        ;0:K/ 1:Y/ 2:M/ 3:C
5EE3:                0605: lpcnt:   DS      1        ;counter(0..3)
5EE4:                0606: lp8dv:   DS      1
5EE5:                0607: ;
5EE5:                0608: lpx:     DS      2
5EE7:                0609: lpy:     DS      2
5EE9:                0610: lpdx:    DS      2
5EEB:                0611: lpdy:    DS      2
5EED:                0612: lpy1:    DS      2
5EEF:                0613: lps1:    DS      1
5EF0:                0614:          ;
5EF0:                0615: LPTLEN:  DS      2
5EF2:                0616: LPTLX:   DS      2        ;LPTLX*pin/8 - LPTLEN
5EF4:                0617: LPTWK:   DS      8        ;プリンタワーク
5EFC:                0618: LGRPDT:  DS      3*12     ;GRBデータワーク
5F20:                0619:          ORG     08000H,08000H
8000:                0620: LPTBF:                    ;プリンタ用バッファ
8000:                0621:
```

▶もうすぐスキーのシーズンだ！　待ちどおしい。白銀の大自然のなかを思いっきり滑り降りるのは本当に最高です。こたつのなかでミカンを食べながらじっくりプログラムやゲームするのも仲間たちと思いっきり遊ぶのも最高。今年はしっかり雪が積もってくれるのを願う毎日です。　杉本　敏光（18）愛知県

DōGA・CGアニメーション講座〈5〉

いぶし銀はどんな色?

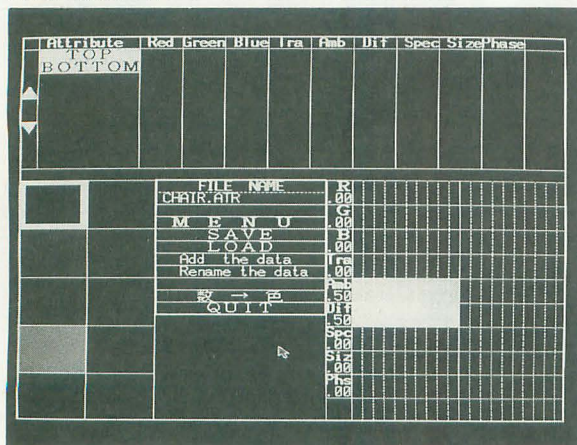
——コラムの逆襲!——

プロジェクトチームDōGA かまた ゆたか

アトリビュートは物体表面の色や材質感を表現するデータです。パラメータの数も限りがありますし，適当に設定してもそれなりのものができるので，初心者の方は軽視しがちです。ですからここにコルようになると，いわゆる“違いのわかる男”と呼ばれるようになるのです。

CGAシステムの応募がいっこうに減りません。たくさんの応募があるのは最初の2週間だけだろうとばかり思っていました。なのに数カ月たった現在もちっとも減る様子がない。だれかこの底なし沼のような作業から助け出してくれ〜(などといっては,せっかく申し込んでいただいた方々に失礼ですね。スミマセン)。

最近，アンケートがたくさん送られてきておりますが，CGAシステムは好評というより，“すごい”とか“おどろいた”という感想がほとんどです。しかしながら，すでに内容などはこの連載で紹介しているのですから，なぜ今さら驚かれるのかちょっとクエスチョンです。私の紹介の仕方が悪かったのでしょうか?

さて，今回はアトリビュートデザインのテクニックを紹介しましょう。正直にいって，このアトリビュートデザインのテクニックは，教えてもらえればすぐに上達するというものではなく，たくさんのカットを制作していく経験の積み重ねによるカンみたいなものです。初心者の方はRGBで色を設定するだけで，他のパラメータはAUTO(自動生成ツール)がつくってくれた値をそのまま利用しておけば，とりあえずはかまわないのですが，そのうち「雰囲気がイメージと異なる」,「リアリティがなく，妙に安っぽい」,「せっかく面の多い形状データをつくったのにのっぺりと同じ色で表示されてしまう」などの不満が出てくるでしょう。そういったとき，今回の解説が解決の糸口にでもなれば幸いです。

ATR(アトリビュートエディタ)の使い方

ATRはマウス操作でアトリビュートのパラメータを設定するツールなのですが，アンケートを見ても，使い方がわからないという苦情はみられないので，特に解説する必要はないでしょう。ただ，「操作性が悪い」とか「RGB以外のパラメータが視覚的に確認できない」などの苦情は多くいただいております。確かにその通りです。現在ATRは大幅に改良中です。開発担当者がいうには各コマンドをウィンドウにしたり，球を表示してハイライトや半透明もリアルタイムに表現するそうです。はて，いつ完成するのでしょうね。

ATRを使用する際の注意事項は以下の通りです。

- 画面下半分は色などの各パラメータを調合するパレットなので，そこで値を設定しても，上半分の数値パネルに移してやらないと，どのアトリビュートにも設定したことにはなりません。
- 数値パネル部は色データ部と材質データ部に分かれています。パレットで設定した値を移すときは必ず両方に移さなければいけません。しかしこれを利用すると，材質は同じだがRGBだけが異なるデータが複数あった場合などに，先に材質部だけ与えておいて，あとからRGBだけを変更することができます。

RBGによる色のデザイン

色のデザインの仕方自体は見ればわかる。やればわかる。ということで，ちょっとしたテクニックを紹介しましょう。

*

小羊：できた絵を見ると，なんかまとまりが悪いというか，いくらRGBを変えてみてもしっくりといきません。どうすればよいでしょうか。

殿：なんかコーナーを間違えているような気もしますが，とりあえずひとつ安易な解決方法を伝授しましょう。

まとまりが悪い原因は色の使いすぎによるものです。

たくさんの色を使うのは悪いことではないのですが，赤，青，黄，緑，紫，白，黒，茶……と無節操に使えばまとまりが悪くなるのは当然です。そこで，使う色を青系統とか赤系統とかパステル調とかの統一性を持たせてやるのです。実際に青系統の色をRGBで設定するには，Rの値は常に0.4以下に抑えるとか，Bの値はすべて0.8以上にするといった規則を設けてやればよいのです。同様にパステル調にするときは，RGBのすべての値を0.5以上にします。

この方法は確かに簡単で有効ですが，一般美術界においてはあまりにも安易なので，禁じ手に近い手法です。ですから，実際の作品をつくるときには，ポイントとなる物体の一部に，わざと規則からはずれた色を設定することでアクセントをつけることがよくあります。

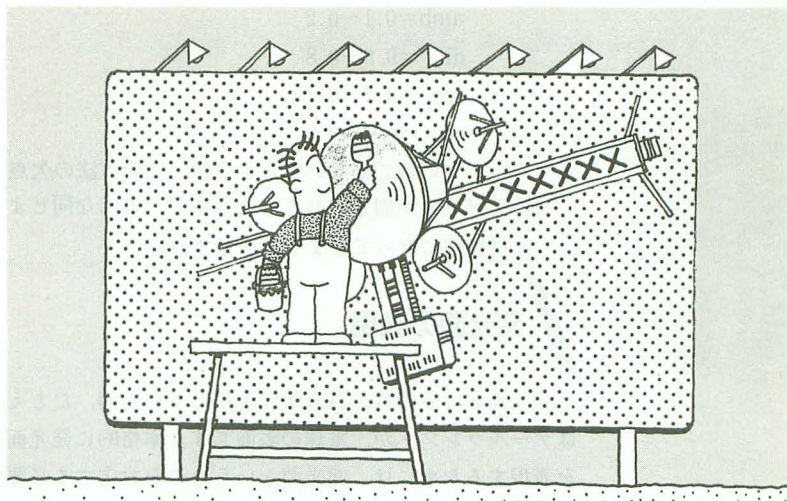
はやくこんな手法に頼らずに，自分の色（または色の組み合わせ）というものを見つけ，オリジナリティのある画面が作れるようになってください。

小羊：色のデザインにおいて注意が必要なことはありませんか。

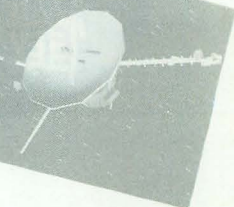
殿：特に注意というほどのものではないのですが，VTRに録画する作品をつくる場合，赤い色に気をつけてください。皆さんご存じでしょうが赤い色はVTRでダビングするときに劣化が激しい性質があります。ですから，イメージユニットでVTR用のNTSC信号に変換した際にもかなりにじんだり，かすれたりします。メインになる物体を赤っぽい色にするのはあまりおすすめできません。また，テールランプなどどうしても赤が必要で，ある程度ポイントになる場合は，色が抜け落ちることを予想して，若干明るく設定しておいたほうがよいでしょう。

しかし逆ににじむ性質を利用すると，火花などはVTRに落として見たほうがそれっぽく見えます。はやくS端子ぐらい付いた高画質の変換機が出てほしいものです。

小羊：せっかく複雑な形をつくったのに，作画するとみんな同じ色になって凹凸がなくなってしまいます。

殿：これには3つの解決方法があります（2，3は色のテクニックとはいえませんが……）。

1．同じ色にならないように，実際に異なる色を配置する………問題の場所の付近の面を同じアトリビュートで指定せずに，3，4種類の少しだけ色が異なるアトリビュートをランダムに指定します。境界線がはっきりして，立体感が出ます。

2．異なる材質にする………1.と同様に複数のアトリビュートを用意し，色はそのままで材質データを少しずつ変えておきます。1.と異なる点は，色が同じなのでそれほど不自然さがないということで，光の当たり方や視点との位置関係が刻々と変化する，つまりアニメーションの場合に特に有効です。

3．アンビエントを小さくする………正統派のやり方です。アンビエントというのは乱反射によるその空間自体の明るさ，光源に関わらず照らされる量なのですが，ひと言でいうと最も暗くなったときにどこまで暗くなるかということです。このアンビエントを下げ，そのぶんディフューズを大きくすると，暗いときと明るいときの差が激しくなって，微妙な凹凸も表現できるようになるわけです。ついでにスペキュラーも若干強めに設定すると，より効果的です。具体的には，

各読者通達事項

DōGA・CGAシステム配付終了のお知らせ

本当にたくさんの申し込みありがとうございました。しかし，あまりにも多い申し込みと，“まだ来ないぞ”という問い合わせに追われて，とうとう受付担当者が夜逃げしてしまいました。そういうわけで，DōGA CGAシステムの受付は原則として，10月31日をもって終了させていただきます。

“そいつは困った”と慌てる方もいらっしゃるでしょう。ご安心ください。これは原則として終了するだけで，どうしてもという方には配付を続けます。受付担当者は“こんな面白くない作業を毎日毎日続けていられない”と言っているのですから，受付の作業が面白くなればよいのです。

11月以降にDōGA・CGAシステムを申し込まれる方は，必ずギャグ，コント，小咄等を同封してください。もちろん実話でも結構です（郵便振込の方は振込用紙の裏面に記入してください）。バカウケした場合，最優先に発送致します。シラケた場合には，あとに回される可能性があります。十分ご注意ください。

ご存じのことと思いますが，大阪の人間は“おもろい”か，“おもろない”かだけを価値基準にしています。

この通達をどこまで本気にするかは読者の皆さんにおまかせ致します。

あて先不明につき発送不能

以下の方ご連絡ください。発送したCGAシステムが戻ってきています。

- 安城市　内海様
- 西春日井郡　岩波様
- 松本市　福井様
- 川崎市　由水様
- 大東市　鵜木様

なかには3セットも申し込まれた方もいるのに……。そのほか各種トラブルで調査中の方がかなりいらっしゃいます。7月，8月に申し込まれて，まだ着かないという方は，TEL，正確な住所，申し込んだ月日を明記してご連絡ください。

amb＝0.1〜0.2
dif　＝0.9〜0.8
spc　＝0.4〜0.7
siz　＝0.1〜0.3

ぐらいがよいのではないでしょうか。この方法の欠点はパラメータが固定されてしまうので，みんな同じような材質感になってしまうことです。

発光面の表現

発光面というのは，自らが光を出している面，たとえばテールランプとか，電球の表面です。本格的に発光面を表現するためには，面光源というものを設定する必要があるのですが，さすがにこのCGAシステムではサポートしていません（CGAシステムでサポートしているのは，複数の点光源と平行光線です)。しかし，この発光面を簡単にそれっぽく表現する方法があります。アンビエントを1.0にすればよいのです。

アンビエントが1.0ということは，光源にまったく影響を受けない，つまり光源が暗くなったりしても，少しも暗くならず，RGBで設定した色そのものになるということです。このことを利用すると，周りが暗くなっても明るいままなので，光っているように見えるというわけです。

Oh!X 8月号の口絵をご覧ください。ライトに照らされるえんぴつと電話がありましたね。この口絵で用いられているさまざまな実験的な手法は，ほとんどが汎用性がありませんのでここではあまり詳しく解説しませんが，電話の右上の緑色のランプにご注目ください。ライトが暗くなろうが，色が変わろうが同じ明るさになっています。ただこれだけのことなのですが，これでちゃんと光っているように見えるから不思議です。この例では，緑のライトの面の周辺に緑がちょっとぼけたような色の面を置いて，よりそれらしくしています（なかなか芸が細かいでしょう)。同様に，Oh!X 7月号の口絵の夕焼け空を飛ぶジェット機の噴射口にもこの手法を用いています。宇宙船の艦橋，夜の新宿に浮かぶビル街，飛行機の警告灯などなど応用はいろいろあります。簡単な手法ですので一度お試しあれ。

また，発光面とは異なるのですが，床のアトリビュートデザインでアンビエントを1.0にすることがよくあります。床にお気にいりの色で模様を入れても，光の当て方によって妙に暗くなっては意味がありません。その場合，光源に関わりなくRGBで設定した色になるという性質が有効になるのです。

コラム 懺悔室

主：このコーナーは愚かなる人間達が悔い改めるために新しく設けられたのじゃ。最初に懺悔するのはどこのどいつじゃ。

R：私はフロッピーのパッケージを制作しております「システムR」と申します。私どもの納品が3週間ばかり遅れたために，CGAシステムの配付が一時中断し，多くの方々にご迷惑をおかけしましたことを深くお詫び致します。

主：なぜ，そのようなことになったのじゃ！

R：はい，台風が当方を直撃し，IFの倉庫が水没してしまったのです。

主：段ボール製のパッケージをそのようなところに保管するとは，不注意きわまりない。血の池で血没するがよい。

R：お許しを〜，ゴボゴボゴボ……。

小林：作画プログラムRENDを開発しました小林です。どうも。実はVer.2.01に大きなバグが発見されました。ときどき法線ベクトルの向きが逆転して明るい面が暗くなったり，暗い面が明るくなったりします。Ver.2.00を改良するときに，エンバグ(バグを加えること)したようです。

主：論外である。Ver.2.01を持っているユーザーに対して，無償のバージョンアップサービスを行え。

小林：おっと。うちの会計が許してくれません。

主：ならば，ちゃんとアンケートを送ってきた正規ユーザーだけでよかろう。

小林：わかりました。Ver.2.01を送った方で，ユーザー登録された方には，年末にでも最終バージョンをお送りします。

主：精進せいよ。

I：フロッピーのフォーマットとコピーをやっております I と申します。当方ではPC-98を利用しているのですが，CGAシステムをコピーするときに関係のないディスクが5枚混入して，気がつかずに発送してしまいました。まことに申し訳ございません。

主：X68000のユーザーに98の26セクタフォーマットのディスクを送るとはまことに失礼な奴じゃ。

I：連絡のあった5名の方には，すぐ新しいディスクをお送りしました。

主：被害も少なく，迅速に対応したことに免じて許してやろう。

小林：PES（制作環境ウィンドウシステム）を開発しました。小林です。

主：またおまえか。今度はなにをした。

小林：先月でも紹介したPESの隠し機能で，画面上のあるポイントをクリックするとメッセージが出るのですが，しつこくそれを続けていると冗談で暴走するようにプログラムしておきました。

主：なに〜冗談ですむか！

リアリティの出し方

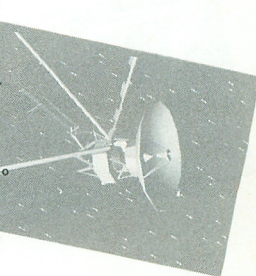
CGによってつくられた映像のイメージというと，まず「きれい」という答えが多いようです。この妙にきれいであることが，逆にCGのリアリティをなくしているといえます。この妙なきれいさは，実際の物体に付いている汚れとか小さい傷まで表現していないから……と思われがちですが私はそうは思いません。また，もしそれが原因であるとすれば，リアリティのあるCGAは本格的なグラフィックワークステーションでしか不可能であるということになってしまいます。なんとかパソコンCGAでもリアリティのある映像を出せないものでしょうか。

リアリティを出すためには，まずモデリングの段階である程度面の数を増やしておかなければいけません。あまりにデフォルメ，単純化した形状ではリアリティが出ないのは当然でしょう。そして，そのように細部までつくったデータを表示するときは，光の当て方や，アトリビュートを工夫して，その細部まで見えるようにするのが人情というものです。実はこの点に問題があったのではないでしょうか。

現実に，ある物体を写真に撮ってよく観察してみてください。飛行機，自動車，科学専門誌に載っている宇宙船。そんな細部まで写っているのでしょうか？　よほど都合のよい光の当て方でもしないかぎり，影になって何も写っていないところや，写っていてもどんな形をしているのかよくわからないところが，かなり多くあります。人間は目で見ているのではなく，脳で認識しているのです。見えないところは，ある程度脳が補ってくれます。

つまりリアリティを出すためには，見えないところを増やして，脳をだましてやればよいのです。なんかマユツバな結論ですね。

具体的に，見えないところを増やすためには，アンビエントを下げ(0.1～0.2)，ディフューズもそれほど上げず（0.6～0.7)，やや逆光になるように光を当てるのです。まったく何も見えなければ，脳もだまされにくいので，スペキュラーはちょっと入れておきます。こうすると，アニメーションの途中の数フレームだけ，少しだけ色の違う小さな面がちらりと見え，あたかもその付近は複雑な形になっているように誤解してくれるわけです。

作品の中の１カットの場合，そのカットだけの問題ではなく，前後のカットも工夫する必要があります。前のカットがぜんぜんリアリティがなければ，脳をだましようもありません。例としては，少なめの星がきらめく真

小林：いえ，暴走に見せかけているだけで，実はESCキーを押すとちゃんと止まります。またさらにもう一度行うとリセットがかかるようになっておりますが，データに害を与えるようなことはまったくしておりません。

主：許さん。次回の発送までに修正するとともに，罰として１年以内に新しいプログラムを開発して発表することを申しつける。わかったな。

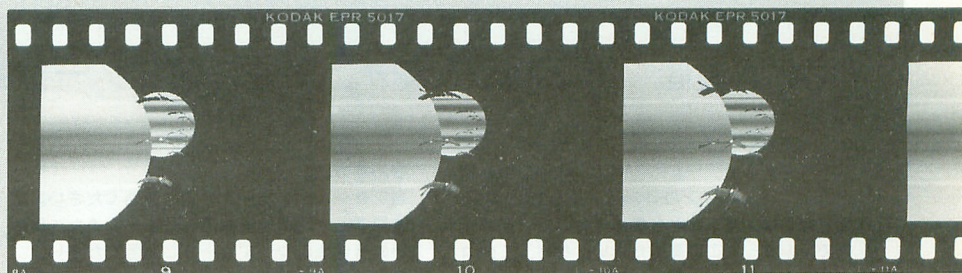
小林：はい，精進します。

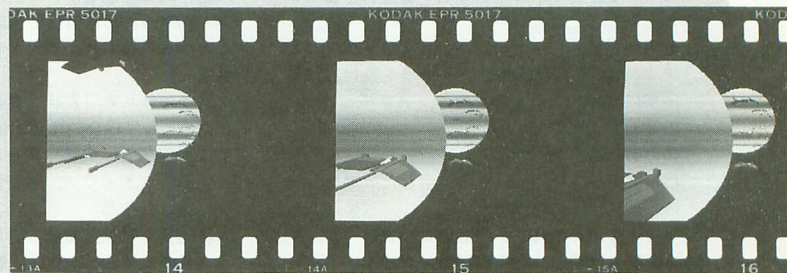
B：まいど，CGAシステムの梱包，宛名書きをやっておりますＢと申します。当社では８月の中旬に10日間の夏季休暇を設けました。その分CGAシステムの発送が遅れてしまいました。

主：夏季休暇とあらば，しかたあるまい。労働者にも人権がある。人権が認められていないのは，DōGAのスタッフぐらいのものだ。だが，どうもこの梱包，宛名書きの作業は時間がかかりすぎているぞ。数が多いのはわかるが，10日以上というのはちょっと遅すぎやしないか？　それに一度に発送できる量が限られているのも，発送が遅れる原因じゃ。

B：すんまへん。

主：それにしても，発送が遅れているようじゃ。責任者出てこい。

かまた：へこへこ。一見，単純に思える発送システムも，読者の皆さんが申し込まれてから，CGAシステムが届けられるまで実際には８つの業者が関与しています。それだけ時間もかかりトラブルも多くなってしまいました。これもすべては配布の実費を最小限に抑えるためです。どうかお許しください。

主：それだけで，こんなに時間がかかるとも思えぬが……。

かまた：へこへこ。もちろん応募の数が予想をはるかに上回るものだったことも大きな原因です。かなり多めに印刷しておいたマニュアルが簡単になくなり，思い切った増刷をしましたが，それももう底をついてしまったぐらいですので，とてもスタッフの手におえません。また，発送業者との契約で，ある程度まとめて出さなければなりません。つまり，不運にも発送した翌日に申し込まれた方は，発送の数が揃うまでかなり待たされることになります。

主：なんかよくわからんが，とりあえずみんなおまえが悪い！

かまた：へこへこ～。

っ黒の宇宙空間をしばらく見せたあとで，ほとんどが影になって形もよくわからないような物を横切らせると，妙にリアリティが出ます（作画時にアンチエイリアスを使うとより効果的です）。

私も先日，夜の大阪の街をビデオカメラで撮影した映像に，「ブレードランナー」に出てくるようなエアーカーをCGAシステムでつくり，スーパーインポーズさせてみましたが，十分見るに耐えるものになりました。さらに，街灯がぽつぽつと並んでいる道を撮影し，その街灯付近に点光源を置いてエアーカーが着陸するシーンをつくってみました。街灯の近くを横切るときに，ほんのりと明るくなり，結構かっこよくできました（ただ，ヤラセすぎで，リアリティは落ちたという意見もありましたが）。

面白いと感じていただければ，チャレンジしてみてください。はっきりいって，いきなりイメージ通りのものはできないでしょう。しかし自分の表現力というものは，試行錯誤を繰り返す過程で生まれてくるものなのです。

おわりに

当然のことなのですが，物体表面の色というのはアトリビュートだけの問題ではなく，光の当て方にも大きく影響されます。いずれ，その方面についても解説する機会を設けたいと思います。

さて，気がつかれた方も多いと思いますが，今月ついに本文よりもコラムのほうが量が多くなってしまいました。誌面を充実させるためにも，読者の皆さんから「あのコラムが面白い」とか「本文なんかもういらない」などのご指導をいただけると大変うれしいので，また送ってください。

次回からはついにモーションデザインに入ります。モーションデザインはCGA制作において最も重要でクリエイティブで奥の深いところです。2，3回に分けて，初級，中級，応用とステップアップしていきましょう。それではまた来月をお楽しみに。

わ〜い，わ〜い，にこ・にこ・ぷん！

★DōGA・CGA システムは一般のお店では取り扱っていません。私達の活動に賛同してくださるアマチュアの方には，カンパ（1口：1000円より）と実費（3000円）だけで配布しています（プログラムは無料です)。郵便振替で申し込んでください。

申し込み期間：1989年7月1日〜10月30日
郵便振替口座：大阪 3-109598　口座名：鎌田　優

または，DōGAプロジェクトルームCGA システム配布係に直接現金書留で申し込んでいただいても結構です。

なお，発送は申し込まれた順番に行いますので，場合によっては多少遅れることがあります。

★CGA システム，本連載，各コラムについてのお手紙お待ちしています。

〒533 大阪市東淀川区淡路5-17-24 篤コーポ102号室
DōGAプロジェクトルーム「なんでもどんとこい」係

DōGA主催　第2回　アマチュアCGアニメーションコンテストのお知らせ

プロジェクトチームDōGAでは，昨年に引き続きCGアニメーションのコンテストを行います。賞金などは，皆さんから集めましたカンパですので，ふるってご応募ください。なお，このコンテストにおいて，DōGA CGAシステムを使用しているかどうかは，いっさい審査に影響しません（念のため）。

●開催主旨
アマチュアのCGA作品の発表の場を設け，広く一般にCGAをPRするとともに，アマチュアCGAの質的向上を促進する。

●募集作品
パーソナルコンピュータを使用したアマチュアのオリジナル映像作品。
＊実写等が含まれていてもかまいませんが，その部分は審査の対象にはなりません。
＊単一の静止画は基本的に不可。
＊プロの方でも，プライベートの機材で，プライベートに制作したものであれば問題ありません。
＊8mmフィルム，またはビデオテープ。

●募集期間
1989年12月31日（必着）

●入賞発表
入選作品の上映会および，表彰式を1990年2月に東京にて行います。入賞者には同時に通知します。また，同年3月には神戸にて上映会を行います。

●応募方法
当プロジェクトチームまでご連絡ください。応募要項と応募用紙をお送りします。また，BGM等の著作権については十分ご注意ください。

●入選作品の使用
入選作品は主催者が行う上映会で使用するほか，CGAのPRのために無償で複製，配布，放送を行うことがあります。

●賞
＊最優秀作品賞　賞状，賞金20万円 ………1点
CGAとして総合的に最も優れた作品。
＊優秀作品賞　賞状，賞金10万円 …………3点
技術，ストーリー，芸術性など何かの点で特に優れた作品。
＊奨励賞　賞状，賞金5万円………………数点
今後のアマチュアCGAに影響をもたらす可能性の高い作品。
＊特別賞　賞状
CGAに貢献した個人，団体，ソフト，ハード，そのほか。
＊特別審査員
・月刊「PIXEL」 編集長　河内　隆幸
・NHK　為ヶ谷秀一
・月刊「ASCII」 編集長　土田　米一
・CGアーチスト　野地　朱真
・月刊「Oh!X」 編集長　前田　徹
（五十音順）

＊

昨年度入賞作品紹介

最優秀作品賞　該当作品なし

優秀作品賞
★「つの虫君の大冒険」　京大マイコンクラブ
つの虫君が野原で不思議な物体を見つけ中に入ってみる。そこにやってきたいじめっこのキョンシーは……。ストーリー性，キャラクター性などを重視しており見るものを楽しませてくれる。
★「PCGA」　梅沢　順
長年にわたって制作したデータを音楽に合わせて再編集したもの。ストーリー等はないが，動きの演出がよく，見る人を飽きさせない。
★「ART-2」　鳥取大学電子計算機研究会
部品が変形して，組み合わさって，ひとつのエンジンが完成するなどの小作品集。パソコンを32台揃え，3カ月間に渡って計算させたという力作。

奨励賞
★「The Fireworks」　大阪府立大学 RANDOM
★「Mの喜劇」　大阪大学コンピュータクラブ

特別賞　CG連合
X68000

＊

●お問い合わせ先
〒533　大阪市東淀川区淡路 5-17-24
篤コーポ　102号室
プロジェクトチーム　DōGA

CGA教養講座 「世界の最先端のCGA作品の動向」その2

《モーションデザインへのアプローチ》

●

前回（といっても9月号）では，レイトレーシングに代表されるレンダリングアルゴリズムの追求が4，5年前から下火になったという話をした。こう書くと最近は新たなアルゴリズムが発表されていないかのような誤解を招きそうなので，今回のテーマに入る前に少し補足をしておく。

たとえば一昨年ぐらいから結構注目を集めているアルゴリズムにボリュームレンダリングというのがある。物体の表面だけに着目するのではなく，詰まっている中身も考えようというもので，流体の密度分布だとか，人体の内臓の動いている様子を表現することができる。どんな絵ができるのかといえば，全体が半透明になって，密度の濃いところが色も濃くなったような感じだ。しかし通常，人体を見るときに内臓まで見える必要はないわけで，使用するケースが非常に限られているのだから，正直いってあまり実用性のあるアルゴリズムだともいえないのではないだろうか。

また，純粋にレンダリングアルゴリズムの問題ではないのだが，自然な人体（特に顔）の表現については，現在も活発に研究されている。昨年メタ・コーポレーション・ジャパンの高沖氏が発表した「メタコス氏の肖像」という作品では，メタボールを使用してかなりリアルな人体を表現して見せた。他の作品も，なかなかよい線までいっているのだが，まだひとつ大きな問題点が残されている。それは1人10万本といわれる“髪の毛”の表現である。だから，現在CGに出演する俳優たちはやたらに“ハゲ”が多い。

さて，レンダリングアルゴリズムの追求に代わって現れた新しい“流れ”は多岐にわたる。そのうちのいくつかを思いつくままに紹介していこう。

まず，ひとつにモーションデザインへのアプローチがある。動きのデザインは作品の出来に直結するきわめて重要な問題にもかかわらず，従来軽視されてきたように感じる。結論から言ってしまえば，今なお特にこれといった手法は発見されていない。

日本が生み出した古典的名作のひとつに大阪大学とリンクスの「BIO SENSOR」がある(大村先生お元気ですか？)。サイボーグの虎がのっしのっし歩き，ピラミッドの上に現れたガイコツの怪物をやっつけるというもので，有名なのでご覧になった方も多いと思う。この虎の動きは，動物園にいって本物の虎を撮影し，それを見ながらアニメータが3面図をおこし，データ入力したそうだ。また，ラストにロボットが走るシーンがあるが，あれはアジス・アベベかだれかの走る姿からデータを作成したそうだ。人のアラさがしをするようで悪いが，虎の前足をよく見ると，ときどきつま先が地面の中にもぐってなくなってしまっている。

その数年後，こんどはアメリカのどこぞの大学から発表された作品では，ワイヤーフレームだが，人が長い布をなびかせながら走っていく様子などを非常に自然に表現してみせた。これは実際の人体に多数の豆球をつけ，暗い部屋で演技してもらった様子を，前後左右から撮影し，データをつくっている。

以上の2つの作品の例では，実際に存在するもの，動きにしか対応することができず，CGの魅力をフルに引き出すことはできない。これらに対して，プログラムによって動きをつくりだそうという試みも当然行われてきた。

1987年にシンボリックス社が発表した「STANLEY AND STELLA : BREAKING THE ICE」（うっ長いタイトルだ）は，鳥と魚が空と海という隔たれた空間で愛し合い，間に存在する氷の壁を突き破るといったストーリーで作品的にも完成度が高い。この作品では実験的行動シミュレーションと称して，鳥や魚の群れの動きをプログラムによって生成していた。簡単に言うと，群れの先頭の動きを与えれば，近くにいる他のものは，障害物を避けながら自動的に追随していくのだ。

同じころアメリカのどこぞの大学（作風が似ているから，上記の大学と同じかもしれない）がやはりワイヤーフレームでつくった作品では，“各関節でのエネルギー消費量を最小限に抑えるという”条件を与えることで，障害物を飛び越える様子を自然にデザインすることができた。この様に，なんらかの制約条件を与えることで，複雑な動きを簡単なパラメータだけでデザインするというのは，パーソナルCGデザインにおいても有効かつ現実的ではないのだろうか。

また，この方向をさらに進めて，物理現象のシミュレーションのような作品も多い。「CALTECH MODELING DEMO 1987」ではぶら下がっているくさりの動きを，「DYNAMIC SIMULATIONS OF FLEXIBLE OBJECTS」では風になびく旗を，「MASAKA TYANTO YONDERU WAKENAIYONA」ではひっくりかえる自動車を表現している。きわめつけなのが「NATURAL PHENOMENA」で，ミミズの動きを見事に表現して多くの者の気分を悪くさせた。このように，花火だとか滝だとかある特定の物に絞って動きを表現することは現在も活発に研究されている。しかし，汎用性があって，手軽に自由に自然な動きをデザインすることはまだまだ難しい。

それではちょっと，日本のアマチュアの作品を見てみよう。1987年コムテックCGコンテスト優秀賞の「AWA」（制作：大阪大学コンピュータクラブ）では，ロボットの集団が阿波踊りを踊っている。これは，関節をつなげるちょっとしたエディタをつくっていくつかのポーズをデザインし，各関節の角度をスプライン関数によって中割りしている。先月号で紹介した「365」もほとんど同じ方法なのだから，このチームが3年間怠けていたことがよくわかる。

また，1989年アマチュアCGアニメーションコンテスト奨励賞受賞の「The Fireworks」(制作：大阪府立大学コンピュータハウスRANDOM）ではプログラムによって美しい花火を表現した。この花火は現実のものをシミュレートしたのではなく，各スタッフが自由にデザインして個性ある動きを実現していた。さらに京都大学マイコンクラブでは，将棋を自由に配置し，1カ所を倒すと自動的に将棋倒しを表現する作品を発表していた。このようにアマチュアレベルでもいろいろな試みは行われている。

残念ながら今年のSIGGRAPHのビデオテープはまだ見ていない（近日中には届くと思う）が，今年もこのような動きのデザインに関する面白い作品があるだろうと期待している。

おっと，もうページがない。どうやらこのコラムはまだまだ続くことになりそうだ。なお，このコラムの内容は，なんの参考文献もなしに，私のうろ覚えだけをたよりに書いているので，ちょっとした思い違いから，うそ八百まで混じっているかもしれませんが，そのへんは，まぁご了承ください。

第32回　知能機械概論――お茶目な計算機たち――

バルセロナの赤い計算機

日本発1週間前

僕は先日，遠い遠い国スペインに足を踏み入れてきました。学生時代もお金を貯めては外国に行ってきましたが，そのたびに大きな影響を受けたものでした。よく，海外旅行から帰ってきて，「やっぱり日本はいい」ということばかりいっている人がいますけれども，僕の場合は，どこに行っても，「旅行者としてではなく，居住者として，もっと長い間いたいものだなあ」と思ってきました。そういう気持ちがまた僕を異国に向かわせるのでしょう。

今回行くスペインは，次回のオリンピックがバルセロナで開かれることもあって，かなりミーハーなような感じがします。しかし，まあそんなこと以前に，どんな国が待ち受けているのかと思うと，ワクワクするのも事実です。

せっかく，日本からずいぶんと離れた国スペインに行くのですから，スペインにある「お茶目な計算機」を探してみようかなと思いました。そこで，ちょっと唐突ですが，勝手にキーワードを作ってみることにしました。「バルセロナの赤い計算機」なるものを探してみようというのはどうでしょう。別に「スペインの青い知能機械」でも，「アルハンブラ宮殿の横に落ちてた電卓」でもよいのですが，まあ感覚的に決めたという，ただそれだけのことです。

モスクワからスペインへ

ついに当日，成田をマドリッドに向けて出発。といっても，切符の都合上，モスクワでの乗り換えが長時間あります。その間中，いろいろなところをうろつき回っていました。といっても，トランジットですから，外に出られるわけではありません。免税店を見物するのが関の山です。

ぼんやりとベンチに座っていると，通路の端や，受け付けなどのところに，（専用の端末以外の）計算機が置いてあります。こわそうなおばさん（失礼！）が使っていたり，あるいはよく見えない位置にあったりしたので，はっきりはわかりませんでしたが，どのマシンにも「Σdata」というマークが付いているように見えました。

単に端末として，中央の計算機にあるフライト情報を表示していただけのようでした。メーカーは案外日本かもしれませんね。ロシア語のキーボードなどを期待したのに残念でした。

モスクワから，スペインのマドリッドに到着し，外国人としての2週間が始まりました。さてこれから，スペイン語を夜な夜な勉強していかなくてはなりません。なんたって自由旅行ですから。

スペイン人の日本人との基本的な違いは，食べたり飲んだりすることに非常に重きを置いているということにあります。典型的なスペイン人の食生活をちょっとだけ紹介してみましょうか。

日本に比べずいぶん遅い朝食をとったあと，11時ごろにおやつを食べます。そして，豪華な昼食。それこそ，フルコースの食事をワインを飲みながら2時間ほどかけてゆっくり楽しみます。日本円にして500円ぐらい払うだけで，その店推薦の日替わり定食が味わえます。定食といっても前菜から，デザートまで付いて，ハウスワインは飲み放題という夢のようなものです！

昼からワインを飲むことはまったくふつうのことでして，もし水がいいという人はビン入りのミネラルウォーターを注文しなくてはなりません。公務員用の食堂の昼の定食にさえワインが付いてきます。

それから，うらやましいことに，1時間ほど昼寝などをしてから，勤務に戻ります。店や博物館でも，昼食を含むこの優雅な時間は休みにしているところが少なくありません。そしてこのあと，おやつ，町のバーで夕食前の1杯，夕食……などと，詳しく書いたらきりがありません。

とにかく飲食を中心とした1日はこのように過ぎていき，そしてまた，朝食（＝「断食明けの食事」と呼ぶそうだ，英語breakfastでもそうだが）へと，食の喜びの無限サイクルは続くのです。

デパートに並ぶ計算機たち

しかし，悲しいことに，常日ごろ，淡泊な食生活を送っている日本人のひとりである僕は，昼こってり食べてしまうと，夜は胃腸が開店休業状態となりがちです。そういうときは，デパートやスーパーにごく簡単な食事を買いに出かけることになります。

立派なデパート（「エルコルテ・イングレス」といったと思う）に行ったとき，電気製品売場をのぞいてみると，思いがけなくもずらりと計算機が並んでいました。

まず，AMSTRADというIBMコンパチがいい位置に並んでいます。値段はまあ日本と大差ないようです。ということは，スペインの人から見れば，かなり高いということになりましょうか。

いかにも「コンピュータ・ミュージック」というような音が人だかりの中から聞こえてきました。その音を作り出しているのは，アミューズメントマシンとしての完成度を誇るAmigaでした。中心では店員に対していろいろと真剣に質問している人がいます（もちろん意味不明）。MSXのゲームカセットもずいぶんと並んでいました。

スペイン広場を走る筆者

純正のIBMマシンも一応ありましたし，日本製のマシンも目立たないところに，案外数多く置いてあります。もちろん東芝のラップトップはあります。あとはキヤノンのワープロ，エプソンのプリンタなどで，すぐに手に入れられそうでした。

ふつうの電器屋にあまり出くわさなかったこともあって，計算機がきちんと見られるところはデパートだけでした。しかし，デパートといっても日本のイメージよりはかなり高級な感じがしましたけれども。

幻の「赤い計算機」

マドリッドからちっちゃな飛行機でグラナダに到着したときに幻を見てしまったのです。空港のビルで荷物の到着を待っていたときのことです。ふと，後ろのほうを見るとそこには，待望の「赤い計算機」がずらりと並んでいるではありませんか。それは，暗めの赤い色をした，見た目のおしゃれなラップトップタイプのマシンでした。

思わず，「あれこそ！」と同行した人に指し示してしまったところで，チョン！　残念ながら，相手の答えが返ってくるまでに，その幻は消えてしまったのです。恥を恐れずに書くならば，エンジ色をしたボディ（いす），同系色で少し照明を反射気味のモニタ画面（背もたれのマット部分）……要するに，単なるしゃれたベンチがずらりと並んでいただけなのでした。僕の視覚認識サブシステムは誤動作してしまったのです。

天才建築家ガウディと計算機

なんとか，スペインでの最後の訪問地，バルセロナに来ることができました。たいしたハプニングもなく，自分でも驚いているくらいです。強いていえば，ある寺院の前の道端で，親子連れの靴磨き屋に強引に靴を磨かれかかって，片足でぴょんぴょん逃げ回ったぐらいのものです。

さてこの街では，ピカソ美術館，ミロ美術館などを始めとして，数多くの芸術家の作品を見て回りました。特に，ここバルセロナの建築家である巨匠ガウディの生み出した数多くの建築物は僕の目に焼き付いて離れません。

ガウディのミラ邸

空中へニョキニョキとと げとげの棒状の屋根が圧倒的なスケールで突き出している聖家族教会，あるいは，誰が見てもその童話的な世界に引き込まれてしまうようなミラ邸（写真），バトリョ邸など。

門外漢が偉そうにして申し訳ないのですが，ガウディの建築物を外見からごくおおざっぱにいうとすると，それは直線の否定であるといえるでしょう。奇抜，あるいはグロテスクとの限界の微妙なところにあるとも感じられる彼の建築物は，自然の中では無限に存在するが，人工物の中では流線形などというあまりに意識的な名前のついたもの以外は忘れ去られつつある，「非合理性の象徴としての曲線」に満ちあふれています。そしてその曲線を多用した建築物群は住む人，あるいは見る人の想像の翼をいくらでも広げてくれるのです。

彼の作品は，実に女性的であるともいえます。予想もつかないような曲線が無数に柔軟に組み合わさって思いもかけないような安定感をかもしだしているのです。

この世を構成するもの，事柄，概念などというものは，男性的なもの，女性的なものという2つを両極にして，思いもかけないほどうまく分類できるものです。試しにいくつかあげてみましょう。

男性的なもの	女性的なもの
論理	直感
意志	感情
右	左
陽	陰

きりがありませんね。最後に，いま述べた直線，曲線という分類も含めておきましょう。

ひとついえることは，原始社会からこの現代社会までずっと，女性的なものから男性的なものへというベクトルに沿って時間は推移してきたということです。しかし，最近になって後戻りではなく逆の方向の成分を持たせようとする動きが強くなってきたと思います（ちょっと舌足らず過ぎますか？）。

ところで，計算機を作る人を計算機アーキテクト（建築家）と呼びます。ガウディのような革新的なアーキテクトによって作られる計算機とはいかなるものなのでしょう？　突飛なアイデアのようですが，実はそんなにすっ飛んだ話ではないと思います。

ファジー，ニューロ，推論，など計算機に関わる最近のキーワードのうちのかなりのものは，この逆方向の流れの中に位置づけられると思われます。計算機も世の中の流れの向きと連動しているのでしょうか？

闘牛場の最前列で

追い求めてきた赤い計算機は，いったんグラナダで幻となりましたが，このバルセロナで確実に僕の中に何かが残ったように感じられます。これはガウディたちの作品のおかげでしょう。

運よく最前列に座れた闘牛場で真っ赤な血を見せつけられ，突然の土砂降りでずぶぬれとなり，半分もうろうとしながらも，「バルセロナの赤い計算機」のイメージは段段と強まっていくのでした。

参考文献

地球の歩き方㉓：スペイン・ポルトガル編，ダイヤモンド社

第41回

猫とコンピュータ
ボクの友だち

Takazawa Kyoko
高沢 恭子

吹いてくる風や街で見かける木々も，すっかり秋一色となってきて，ちょっぴり肌寒くなってきた今日このごろ。キョウコさんの家でも調子を取り戻したホンニャアが元気に“紙”と格闘しているようで……。

ここちよい秋の日には，どんな小さな音でも庭のすみずみまで響きわたるように思えた。そんなときは植え込みの下から，ホンニャアの真っ白い足がよく見え隠れしたものだ。白い足の歩く音は聞こえないのだけれど，小さな重みの感触は，ひとつひとつ，澄んだ光をとおりぬけて窓辺まで伝わってきた。あたりはひんやりした深い静けさがあり，それは空まで広がっている。ホンニャアが育ったS市の庭である。

この東京のちっちゃな庭も，しっとりした光と影で彩られるころになると，ホンニャアの白い足が庭木の根もとにあらわれそうな予感がする。彼がいま昼寝をしていようが，ライバルのリサーチに出かけていようが，あの純白の軽い4本足が音をたてずに動いているような気がしてくる。

白い足は，時にかすかな物音にピクリと反応して，まっしぐらにこちらに向かってくることがあった。そう，白い足と澄みわたった庭からのもうひとつの予感は，紙の音だ。

ときめき

ホンニャアが紙というものに対して示す態度や反応は，猫としてはごくふつうのこと，言ってみれば猫に標準装備されている感覚なのかもしれないが，あのただごとでないこだわりはやはり特徴的だと思う。ホンニャアは「紙」をほっておけなかった。

紙といっても日常ほんとにたくさんの紙がある。書く紙，描く紙，読む紙，知らせる紙，包む紙。大きい，小さい，厚い，薄い，硬い，柔らかい紙。光沢も肌ざわりもさまざまの紙たち。

ホンニャアのまわりにも同じようにいろいろな種類の紙があって，わが家はどちらかといえば紙類が多いほうなのかもしれなかった。そんなたくさんの紙たちも，ただ置かれているだけの平らな紙や，お行儀のよい美しい紙はホンニャアにとってはなんの関心も呼びおこさない，死んだ紙だった。

問題なのはそれがひとたび何かのかたちを見せて盛り上がったり，シワになったりして動きはじめた時なのだ。同時におこる大きな要素として，紙の出す音があった。

ともかくホンニャアは紙の音を聞きつけてやってくる。とくに少し面積の大きめの紙の音に心がおどるらしかった。でも新聞紙のようなわりあい柔らかい紙などはホンニャアも穏やかな気分のようで，あまり激しい動きは見せない。読む人がページをめくるときにできるなだらかな屋根の下に，這うようにもぐり込んだり，逆に読んでいる面の上に堂々と寝そべったりした。

新聞紙よりもう少しハリのあるもの，たとえばデパートの包装紙や，配達されたものがくるまれているクラフト紙などはたいへんだった。包みを開くときから，厳密に言えばホンニャアの経験上包みが開かれる前から，騒ぎが始まる。彼の顔の前で大きく右，左に動く紙の踊り，その高らかな音，考えるだけでホンニャアの胸も高鳴ってくる。そして飛びつく。紙は無抵抗に動きながらホンニャアにおおいかぶさり，彼に戦慄をおぼえさせる。しぜんにツメが出て，ホンニャアは紙の中をめまぐるしく走り回るけれど，敵をおさえ込むことはできない。紙の音はホンニャアの動きでいちだんと大きくなるので，興奮はとどまるところがなくなり，やがて紙のあちこちに破れができてくる。

こっちもはじめのうちは，紙の中の奇怪な生き物を見る面白さで見物しているけれど，だんだんおさまりがつかなくなるので恐ろしくなってきたりもするのだ。むろん手なんか出したら，予想以上の深い傷を負わなければならない。

なつかしのBITQUEEN

紙袋の場合は立体と空間がホンニャアをいつも誘惑した。もともとハコや買い物カゴや段ボール箱などはかならず入ってみるのが猫の習性らしく，ホンニャアもこれを義務のように思っているから，たちどころにもぐり込んで，紙との交流をする。

横たわった紙袋のいちばん奥にうずくまり，ゆらゆら動く入り口あたりに攻撃をしかけてくるのだ。こんなとき近くを通りかかった足は、やはり痛烈なヒッカキの急襲を受けることになる。

紙はホンニャアの探究のテーマのひとつであり，格闘の相手だったようだ。その証拠に自分のからだより小さい紙の動きについてはそれほどの関心がなかった。便せんやトレーシングペーパーがいくら音をたてても耳がピクリと動く程度，丸めた上質紙をホンニャアの顔の前にころがしてみてもちょっとおあいそに手を出してみるくらいだった。ホンニャアはいつも耳で紙の音をとらえながら，挑戦相手の選別をしていたにちがいない。

小さい紙でもセロハンの音だけは特別でそれは攻撃の相手としてでなく，好物への連想だった。ホンニャアがいちばん好きな海苔も、お酒のおつまみ類も，みんなセロハンの袋に入っていたから，チクリとでも音がしようものなら大騒ぎ，彼に気づかれないように私たちがそれらを食べるなんてホンニャアの外出時以外あり得ないことだった。

ホンニャアと紙といえばいまでこそ混乱もなく遂行できるようになったのが，プリントアウトの作業である。わが家の初代のプリンタは，ずいぶんと騒々しい音をあげるBITQUEENという機種だった。夏の夜などに，窓を開け放ってこれに仕事をさせ

ようとするのは，とても勇気がいった。

ホンニャアはこの音は大の苦手だったのだが，ゴソゴソと紙がいっしょに動いているのが気になってしようがなかった。遠のいては近づきをくりかえして，ついに手を出すようになってしまった。誰かが動かしているようすもないのに，紙が波うちながら移動していくのだから，ホンニャアにとってこんなに興味のあることはなかったのだろう。BITQUEENが動きだすとどこからかやってきて，プリンタ用紙の箱の中に入ってひっかきまわすので，仕事はやりなおしになることがよくあった。

あれから数年のあいだにプリンタは5，6台入れ替わり，ホンニャアもこればかりはひどく叱られるので態度を改めるようになったのだ。

逃げるエンピツ

紙はホンニャアにとって，独特の動きと音を出す遊び道具であるらしい。いろいろなかたちになるし，ホンニャアが触れることでまた音をたてたり，かたちを変えたりの反応をする。動く紙はホンニャアにとっては生き物で，いっしょにスポーツをする仲間なのだろう。

私たちにとっても紙はなかなか心地よいものだと思う。おおまかに考えて，紙の肌ざわりは優しいし，目的によってこれだけ変身してくれるものも少ないのではないだろうか。

ホンニャアはおもに音と動きを愛しているらしいが，私たちは日用雑貨としてのほかに，なによりも記録や伝達，表現のための必需品としての紙に，一種の敬意をはらっている。

ついこの間までは，印刷活字のたぐいが使われている紙類は，みんな外部や他者からの伝達だった。書籍，教科書，新聞，案内，書類，チケット。そういうものはみんな一方的な押しつけの紙たち。ところがそうでないものは，自由な作業がほどこせる嬉しい紙だった。

白い紙に何かを表現しようとするときの期待のひとときは素晴らしい。紙には道具との協力作業があって，その双方を選択する予測の楽しみがまた大きい。和紙と毛筆，画用紙とパステル，キャンソン紙と木炭というようなおおざっぱな分け方のなかに，さらにこまかな用具と紙との相性の違いがあって，その語り合いがたくさんの変化を見せてくれる。

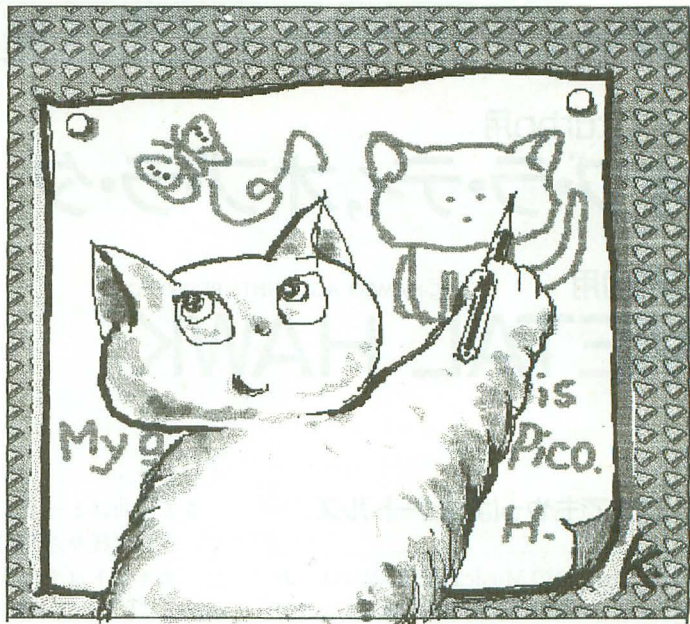
こどものころから，ザラ紙やわら半紙，クロッキーブックのような素朴な紙と，B2かBくらいの鉛筆の取り合わせが好きで，字であれ絵であれ，気にいって使っていた。ちょっと硬めの紙と鉛筆になると，とたんに意欲をなくしたものだ。

画学生のころにケント紙を扱わなければならなくなってとても苦労した。これはもともと印刷や図版に適した紙だから，ぼかしやにじみなどあいまいなものを受けつけるのは上手でない。デジタル表現にふさわしい紙なのだ。カットなどをアルバイト的に描くうちになんとかつきあえるようになってきたけれど，いまだに苦手である。

ホンニャアでなくても，紙はさわるときや何かを書くときの感触や，用具とのなじみぐあい，音色がいろいろに楽しめる。

ところがこのところ，私の日常で紙との語り合いがすこし減ってきてしまった。やっぱりいうまでもなく，パソコンを原点にした印刷現象の芋づる式なだれ込みのためだ。印刷物はよそからくるものだけでなくいますぐ自家製でつくれるようになってしまったのだ。手書き文字への執着が，短期間にあっけなくワープロ汚染されたのを筆頭に，ほら，イラストまでパソコンで描いちゃっている。紙は用具や私との相性でなく，印刷インクとの調和の問題になって，印刷美術の分野になってきた。ザラ紙に鉛筆のナマの感触やケシゴムで消えるものは遠のいてしまった。

興奮のショウ

ホンニャアと紙との極めつけは，障子紙だった。S市のホンニャアの家は，アルミサッシがふつうの大きさの1.5倍くらいはある特注のもので，その内側はカーテンのかわりに障子が入れられていた。

ふだんはピンと張られた障子紙に特別の興味もなくイタズラもしないホンニャアだが，お天気のよい日に庭の木の葉の影がゆらめいたり，小虫がまぎれ込んで飛び回ったりすると事態は一変する。とびつき跳ねまわり，ひっかき，当然障子は掻き傷だらけになるのだけれど，その障子紙の破れる音のなんともいえない心地のよさ，爽快さがホンニャアの動物の本能を奮い立たせるらしかった。

晴れた秋の日に障子を張り換えようとするときは，わが家はそろって緊張した。そしてひそかに期待した。障子を張り換えるには，張ってあるものを取り去らなくてはならない。まずひと刺し，障子の紙を破く音がこの澄んだ庭に響きわたれば，ホンニャアは矢のようにとんできて障子の枠に飛びつき，狂ったようにこれと長い時間格闘をつづける。障子のサンも特注で大きかったから，ホンニャアは自分で破いたところをラクラク通過できた。これはホンニャアとほかのどんな紙とで見せるショウよりも白熱したものだった。私たちはこれを予想してひそかにワクワクしたのだ。

でも問題は後半の張り替えの作業だった。ホンニャアがいるかぎり，新しい障子紙を広げて張り換えることはできないし，全部張り終わるまでの時間ホンニャアに気づかれないのは，ナマやさしいことではなかったからだ。

X1/X1turbo用
オブ・ラ・ディ,オブ・ラ・ダ

Manabe Mitsuo
真鍋 光男

X68000用

METAL HAWK

Shindoh Noriyuki
進藤 慶到

秋も深まってきました。巷では学園祭の時期とあってポップミュージックが溢れかえっています。そんなわけで，今回はあえてポップスを取り上げずに，オールディーズとゲームミュージックで構成してみました。秋の夜長にぜひじっくりと聴いてみてください。

いまでもやっぱりビートルズ

X1/X1turboのMusic BASIC用には，あのビートルズのナンバーから「OB-LA-DI, OB-LA-DA」をお届けします。

「♪オブラディ・オブラダ・パパパーヤ」（このフレーズ以外のところをまともに唄える人を私は知らない……）というサビはおそらく誰もが知っていることでしょう。しかし，ビートルズの曲ということは意外と知られていないようです（現に編集部の中にもどこかの国の童謡だと思っていたヤツもいた）。

私は運動会でこの曲がかかっていたという，かすかな記憶があるのですが，それは本当のことだったのでしょうか？ もし「私も運動会で聞いたことがあるぞ」という確かな記憶がある人はぜひ編集部にご連絡をください。

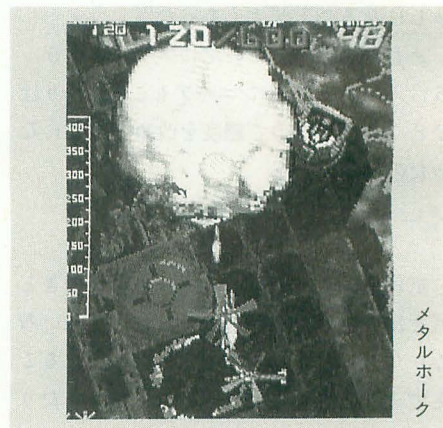
若き日のビートルズ

さて，曲はとっても楽しい仕上がりなので，思わず繰り返し聞きこんでしまいました。ポイントはやはりメロディのBell&Fluteが，とてもよく曲の明るさとマッチングしていることでしょうか。さらにビートルズの曲という選曲の素晴らしさと，全体的なバランスがよかったということで，今月のMVP（Most Valuable Playing data）はこの曲に決定したいと思います。パチパチパチ。

目標X68000・発進

「目標選択・目標確認・メタルホーク・発進！」でお馴染み（？）のシューティングゲーム，メタルホークのミュージックです。このプログラムは派手なシンセパートが売り物で，今どきハヤリのOPMAに対応しています。「巨大なプログラムだからきっとボツでしょう」などという，謙遜とも自信ともとれるようなコメントが一緒についていましたが，君はまだまだアマイ。Oh!Xはヤルときゃヤル雑誌です。たとえメガバイト（！）の単位でも載せるときは載せます（30人以上が絶対入力するのならというのが条件らしい）。

ということで，掲載おめでと～になってます。（編注：イカ天の見すぎ）

このメタルホークってヤツは，よっぽど人気が高いのか，ほんの1週間程度の間で4つも作品が届きました。しかもぜ～んぶX68000・OPMA用というのだからスゴイ。そんななかでもっとも素晴らしかったのがこの作品でした。入力(うち)ごたえもしっかりありますが，なによりそれ以上に聞きごたえがあります。音取りをすべてCDから行ったそうですが，残念なことに若干の違いがありました。ここまでよくできているのなら……，ということで，作者の進藤君の了解を得たうえで，修正をさせていただきました。もちろん，作品のノリは100%もとのままです。

メタルホーク

さあて，音楽の移植も終わったことですし，どなたかゲームのほうも移植してください。出来上がったらきっちり投稿してみてくださいね(おいおい)。　(S.K.)

リスト1 オブ・ラ・ディ,オブ・ラ・ダ

```
10 '
20 '    OB-LA-DI  OB-LA-DA !!    by  BEATLES
30 '                             programed by  M'Factory
40 CLEAR &HFBB0
50 DEFINT a-z:DEFSTR p:DIM p(13)
60 SCREEN
70 CLS 0:TEMPO 0
80 "so"
90 PLAY "xt122";
100 '
110 GOTO "play"
120 '
130 LABEL "wr"
140 '
150 READ n:IF n=999 THEN RETURN
160 PLAY p(n);
170 GOTO 140
180 '
190 LABEL "play"
200 '
210 ' TR 1
220 '
230 p(0)="i1v13o5q8k3p3L16=2h2s2,4,0,3
240 p(1)="f8fff8fff8e8f8d8_1=2
250 p(2)="L8rdrdrdrd
260 p(3)="rcrcrcrc
270 p(4)="re-re-re-re-
280 p(5)="rdrdrcrc
290 p(6)="rcrcrdrd
300 p(7)=p(2)
310 p(8)="cccc<b-rb-4&b-1
320 '
```

```
330 RESTORE 350:"wr":PLAY ":";
340 '
350 DATA 0,1,2,2,2,2, 2,3,3,2,2,4,5,2,2,6,5,2,2,6,5,2
360 DATA              2,3,3,2,2,4,5,2,2,6,5,2,2,6,5
370 DATA 2,4,4,2,2,4,4,2,3
380 DATA              2,3,3,2,2,4,5,2,2,6,5,2,2,6,5
390 DATA 2,4,4,2,2,4,4,2,3
400 DATA              2,3,3,2,2,4,5,2,2,6,5,2,2,6,5
410 DATA 7,7,8,999
420 '
430 '
440 ' TR 2
450 '
460 p(0)="i1v14o5q8k0p1L16=2h2s2,4,0,3_1
470 p(1)="c8ccc8ccc8<b8>c8r8_1=2
480 p(2)="L8o4rb-rb-rb-rb-
490 p(3)="rararara
500 p(4)=p(2)
510 p(5)="rb-rb-rara
520 p(6)="rararb-rb-
530 p(7)=p(2)
540 p(8)="aaaafrf4&f1
550 '
560 RESTORE 350:"wr":PLAY ":";
570 '
580 ' TR 3
590 '
600 p(0)="i1v14o4q8k0p2L16=2h2s2,4,0,3_1
610 p(1)="a8aaa8aaa8g+8a8r8_1=2
620 p(2)="L8rfrfrfrf
630 p(3)=p(2)
640 p(4)="rgrgrgrg
650 p(5)=p(2)
660 p(7)=p(4)
670 p(8)="ffffdrd4&d1
680 '
690 RESTORE 350:"wr":PLAY ":";
700 '
710 ' TR 4
720 '
730 p(0)="i1v13o2q8k3p3L8r1
740 p(1)="o2b-r>df<b-r>df<
750 p(2)="o2b-r>df<b-r>df<
760 p(3)="o2fra>c<fra>c<
770 p(4)="o3e-rgb-e-rgb-
780 p(5)="o2b-r>df<fra>c<
790 p(6)="o3frdfgrdc
800 p(7)="o2b-r>df<b->cc+d
810 p(8)="o3"+STRING$(4,"e-<b->")
820 p(9)="o3"+STRING$(3,"e-<b->")+"ef
830 p(10)="o2"+STRING$(4,"b-f")
840 p(11)="o2"+STRING$(4,"b-f")
850 p(12)="o3"+STRING$(6,"rf")+"re-dc
860 p(13)="o3"+STRING$(4,"grdr")+"fe-dc<b-4r4r1
870 '
880 RESTORE 900:"wr":PLAY ":";
890 '
900 DATA 0,1,1,2,2, 2,3,3,2,2,4,5,2,2,6,5,2,2,6,5,2
910 DATA            2,3,3,2,2,4,5,2,2,6,5,2,2,6,5
920 DATA 7,8,9,10,11,8,9,12
930 DATA            2,3,3,2,2,4',5,2,2,6,5,2,2,6,5
940 DATA 7,8,9,10,11,8,9,12
950 DATA            2,3,3,2,2,4,5,2,2,6,5,2,2,6,5, 13,999
960 '
970 '
980 ' TR 5
990 '
1000 p(0)="i2v14o3q8k0p3L8r1_1
1010 p(1)="o3b-r>df<b-r>df<
1020 p(7)="o2b-b->df<b->cc+d
1030 p(10)="o2"+STRING$(3,"b-f")+"b->f<
1040 p(11)="o2b-fb->f<b->cc+d
1050 '
1060 RESTORE 900:"wr":PLAY ":";
1070 '
1080 ' TR 6
1090 '
1100 p(0)="i5v16o3q8k5p3L64r1r1r1
1110 p(1)=STRING$(2,"i5o3e&e-&d&d-r8.i4d-8r8")
1120 p(2)="i5o3e&e-&d&d-r8.i4d-8r8i5o3e&e-&d&d-r16i4L16d-d-d-d-d
-8L64
1130 p(3)="i5o3e&e-&d&d-r8.r2.r1
1140 p(4)="i5o3e&e-&d&d-r8.i4d-8r8i5o3e&e-&d&d-r8.r4r1
1150 '
1160 RESTORE 1180:"wr":PLAY ":";
1170 '
1180 DATA 0,1,1, 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 :' 1ハ 2+16 コ
1190 DATA        1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,  1,1,1,1,1,1,1,3
1200 DATA        1,1,1,2,1,1,1,1,1,1,1,1,1,1,1,  2,1,1,1,1,1,1,3
1210 DATA        1,1,1,2,1,1,1,2,1,1,1,1,1,1,1,  1,1,4, 999
1220 '
1230 '
1240 ' TR 7
1250 '
1260 p(0)="i6v15=3s2,1,0,5o5q7k0p3L8r1r1r1r1r1=2h3s2,4,0,11
1270 p(1)="o5dddddc<b-a>c4c&c4re-&e-e-e-e-e-e-dc
1280 p(2)="<b-4r4r4r>f&ffffffe-d
1290 p(3)="e-f4g&ggfe-dde-dce-dc<
1300 p(4)="b-4r4rb->df&f<b->df&f<b->df&
1310 p(5)="f4.b-&b-4rf&fe-de-dc4<b-&
1320 p(6)="b-4r2.
1330 p(7)="b-4r>d16&c16&<b-4r4
1340 p(8)="rb-16b-16>e-fg4fgb-re-gr<b-4>d&
1350 p(9)="d4r4r2<
1360 p(10)="r2i1v11o4_4r8a-8r8a-8i6v15o4
1370 p(11)="rb-16b-16>e-fgrfg
1380 p(12)="b-re-g&g4r<b->ddde-&e-d4d&c4r4r2
1390 p(13)="b-4r2r16>g16g16g16b-b-g4r4rgfe-d16d16c<b-4r4r1
1400 '
1410 RESTORE 1430:"wr":PLAY ":";
1420 '
1430 DATA 0, 1,2,3,4,5,4,5,6, 1,2,3,4,5,4,5, 7,8,9,10,11,12
1440 DATA                     1,2,3,4,5,4,5, 7,8,9,10,11,12
1450 DATA                     1,2,3,4,5,4,5, 13, 999
1460 '
1470 '
1480 ' TR 8
1490 '
1500 p(0)="i6v11=3s2,1,0,5o5q8k3p3L8r8r1r1r1r1r1
1510 p(10)="r1"
1520 '
1530 RESTORE 1430:"wr":PLAY ":";
1540 '
1550 ' TR 9
1560 '
1570 p(0)="=1s0,0,0,0^1k0v13L1'rrrrr
1580 p(1)="rrrr4o6_3d8<b-8&b-4r4¯3'
1590 p(2)="rrrr4o6_3d8<b-8r2r¯3'
1600 p(3)=STRING$(7,"r")
1610 p(4)="rrL8o5^2_1dd4dcc4c<b-b-4b-a-a-4a-g1L1^1¯1'rrr
1620 p(5)="rrrr8L8o3b->d4fg4b-&b-b->c4d<b-4>c&c<b-a4b-g4<b-&b-b-
>d4ff4<b-&
1630 p(6)="b-b->d4fg4b-&b-b->c4d<b-4>c&c<b-a4b-g4<b-&b-b->d4ff4<
b-&b-4r2.L1
1640 p(7)="b-4r2.rro4b-8>d16f8d16<b-8>b-8>d16d8d16<b-8
1650 p(8)="ro5e-8g16b-8g16e-8>e-8g16b-8g16e-8 rr8L8o3b->d4fg4b-&
b-b->c4d<b-4>c&c<b-a4b-g4<b-&b-b->d4ff4<b-&
1660 p(9)="rv10o4^2L8e-e-e-e-'r2r1
1670 p(13)="rrL8o5^2dd4dcc4c<b-b-4b-a-a-4a-g1r1ro5^1c<b-agfe-dc<
b-agfe-dcL1
1680 '
1690 RESTORE 1710:"wr":PLAY ":";
1700 '
1710 DATA 0,1,2,3, 1,2,3,4, 1,5,6,13, 7,8,6, 9, 999
1720 '
1730 '
1740 ' TR 10
1750 '
1760 p(0)="=1s0,0,0,0^1k3v9L1'r16rrrrr
1770 p(4)="rr2.r8.L8o4v12^2ff4fe-e-4e-dd4dcc4c<b-1L1v9'^1r16rrr
1780 p(9)="r2.r8.v10o4^2L8e-e-e-e-'r2r1
1790 p(13)="rr2.r8.L8o4v13^2ff4fe-e-4e-dd4dcc4c<b-1v9'r16r1ro5c<
b-agfe-dc<b-agfe-dcL1^1
1800 '
1810 RESTORE 1710:"wr":PLAY ":";
1820 '
1830 ' TR 11
1840 '
1850 p(0)="y7,28L16'r1r1r4."+STRING$(6,"y11,0y12,6y10,16y6,25y13
,0r8")+"=1^1v14
1860 p(1)=STRING$(7,"y6,2br")
1870 p(2)=STRING$(8,"y6,2br")
1880 p(3)=STRING$(5,"y6,2br")+"r4.
1890 p(4)="v15y6,1b4.v14"+STRING$(5,"y6,2br")
1900 p(5)="r1r2v15"+STRING$(4,"y6,25br")+"v14
1910 '
1920 PLAY p(0);:PLAY p(1);
1930 FOR j=1 TO 17
1940 PLAY p(2);
1950 NEXT:sb$=p(3):p(3)=p(2)
1960 FOR l=1 TO 2
1970 FOR j=1 TO 15
1980 IF j=4 THEN PLAY p(3);:GOTO 2000
1990 PLAY p(2);
2000 NEXT
2010 PLAY p(3);:PLAY p(4);
2020 PLAY p(2);:PLAY p(2);:PLAY p(2);:PLAY p(2);
2030 PLAY p(2);:PLAY p(5);
2040 p(3)=sb$
2050 NEXT
2060 FOR j=1 TO 17
2070 IF j=4 OR j=8 THEN PLAY p(3);:GOTO 2090
2080 PLAY p(2);
2090 NEXT
2100 PLAY p(3);:PLAY "r1x"
2110 '
2120 END
2130 '
2140 LABEL "so"
2150 MEM$(&HB190,36)=HEXCHR$("F2 00 71 4D 33 41 23 2D 26 00 5F 9
9 5F 94 05 05 05 07 02 02 02 02 10 10 10 A4 00 00 00 00 00 00 80
 00 00 00"):'01 Gra Piano
2160 MEM$(&HB1B4,36)=HEXCHR$("E0 00 36 35 30 31 19 37 13 00 DF D
F 9F 9F 04 06 09 06 07 06 06 08 29 19 19 F9 00 00 00 00 00 00 80
 00 00 00"):'02 E. Bass
2170 MEM$(&HB1D8,36)=HEXCHR$("DB 00 46 45 43 40 29 07 07 00 1F 1
F 1F 1F 00 07 14 12 00 C8 00 C0 FF 87 FF FA 00 00 00 00 00 00 80
 00 00 00"):'03 Hi Hat close
2180 MEM$(&HB1FC,36)=HEXCHR$("FC 00 0C 01 00 01 00 07 25 00 1F 5
C 5E 9C 00 91 11 8F C0 00 80 00 01 F8 D9 F7 00 00 00 00 F4 C8 80
 00 02 00"):'04 Snare Drum
2190 MEM$(&HB220,36)=HEXCHR$("FE 00 4F 40 30 70 00 06 00 00 1F 1
F 1F 1F 0E 15 11 10 85 D3 09 08 06 88 87 77 00 00 00 00 00 00 80
 00 00 00"):'05 Bass Drum
2200 MEM$(&HB244,36)=HEXCHR$("E4 71 0C 04 01 01 1B 2D 25 23 7f 5
F 14 4E 10 8A 00 88 C0 00 00 00 F8 F5 0A 18 80 00 80 00 00 C6 86
 14 02 00"):'06 Bell&Flute
2210 RETURN
```

リスト2 メタルホーク

```
  10 /*     save "METAL_HAWK        .bas"
  20 /*
  30 /*       M E T A L   H A W K
  40 /*
  50 /*        (C) n a m c o
  60 /*
  70 /*   P R O G R A M E D   B Y   E N G
  80 /*
  90 m_init()
 100 key  3,"          @M
 110 key  8,"trfree()@M
 120 key  9,"m_stop()@M
 130 key 10,"m_play()@M
 140 /*
 150 str p(30)[256]:char o(255),v(4,9),voi(4,10)
 160 str k[256],l[256],m[256]
 170 str hl[256],ml[256],ll[256]
 180 str hs[256],ms[256],ls[256]
 190 /*
 200 for i=1 to 8
 210  m_alloc(i,5000)
 220  m_assign(i,i)
 230 next
 240 /*
 250 vdata()
 260 ddata()
 270 MUS1():MUS2():MUS3()
 280 m_play()
 290 end
 300 /*
 310 /*  S E T   M M L   T O   T R A C K
 320 /*
 330 func t(tt)
 340  n=-1
 350  repeat
 360   n=n+1
 370   m_trk(tt,p(o(n)))
 380  until o(n)=30
 390 endfunc
 400 /*
 410 /*  V O I C E   S E T
 420 /*
 430 func set(vn)
 440  voi(0,0)=(v(4,1)*8)+v(4,0)
 450  voi(0,1)=15
 460  voi(0,9)=3
 470  for x=0 to 3
 480   for y=0 to 9
 490    voi(x+1,y)=v(x,y)
 500   next
 510  next
 520 m_vset(vn,voi)
 530 endfunc
 540 /*
 550 /*  ポルタメント
 560 /*
 570 func pol(pa,pb,pc,pd,pe)
 580   str oto(11)[2]={"c","c+","d","d+","e","f","f+","g","g+",
"a","a+","b"}
 590   k="y"+str$(47+pd)+"," : l="" : x=pa+pb
 600   for i=1 to pc
 610     x=x-pb
 620     if x<0   then { x=256+x
 630       if pe=0  then pe=11:m=">" else pe=pe-1
 640     }
 650     if x>255 then { x=x-256
 660       if pe=11 then pe=0 :m="<" else pe=pe+1
 670     }
 680     l=l+k+str$(x)+m+oto(pe)+"&":m=""
 690   next
 700   l=left$(l,len(l)-1)
 710 endfunc
 720 /*
 730 /*  D R U M   D A T A
 740 /*
 750 func ddata()
 760 hl="o3{e&e-&d&d-&c&>b&b-&a&a-&g}6o4
 770 ml="o2{b&b-&a&a-&g&g-&f&e&e-&d}6 o4
 780 ll="o2{g&g-&f&e&e-&d&d-&c&>b&b-}6o4
 790 hs="o3{e&e-&d&d-&c&>b&b-&a}o4
 800 ms="o2{b&b-&a&a-&g&g-&f&e}o4
 810 ls="o2{g&g-&f&e&e-&d&d-&c}o4
 820 endfunc
 830 /*
 840 /*  V O I C E   D A T A
 850 /*
 860 func vdata()
 870 /*
 880 /*   AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2  BASS
 890 v={  31, 10,  7,  8,  2, 33,  0,  0,  7,  0,
 900      31,  8,  8,  7,  5, 23,  3,  7,  7,  2,
 910      31,  5,  6,  7,  1, 37,  0,  0,  3,  0, /* CON FBL
 920      31,  8,  6,  6,  5,  1,  0,  1,  7,  0,       2,  7}
 930 set(75)
 940 /*   AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2  CHORD1
 950 v={  24, 12,  3,  0,  1, 27,  0,  8,  3,  0,
 960      24, 11,  0,  5,  1,  0,  1,  4,  7,  0,
 970      24, 12,  3,  0,  1, 28,  0,  8,  7,  0, /* CON FBL
 980      24, 11,  0,  5,  1,  0,  1,  8,  3,  0,       4,  6}
 990 set(70)
1000 /*
1010 /*   AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2  MAIN
1020 v={  25, 15,  0,  0,  1, 16,  0,  8,  3,  0,
1030      22,  0,  0, 10,  0,  6,  0,  4,  3,  0,
1040      22,  0,  0, 10,  0,  5,  0,  4,  3,  0, /* CON FBL
1050      29,  6,  7, 10,  3,  3,  0,  3,  3,  0,       5,  0}
1060 set(71)
1070 /*
1080 /*   AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2  GLO
1090 v={  31, 18, 19,  3,  9, 31,  0,  4,  7,  0,
1100      21,  0, 14,  8,  0,  1,  0,  4,  7,  0,
1110      31, 18, 19,  3,  9, 32,  0,  8,  3,  0, /* CON FBL
1120      31,  0, 14,  8,  0,  1,  0,  4,  3,  0,       4,  3}
1130 set(72)
1140 /*
1150 /*   AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2  WHISTLE1
1160 v={  15, 18,  0, 11, 12, 46,  0,  4,  3,  0,
1170      16, 18,  0, 11,  2,  1,  0,  8,  3,  0,
1180      18, 18,  0, 11,  7, 41,  0,  2,  7,  1, /* CON FBL
1190      14, 18,  0, 11,  2,  2,  0,  8,  7,  0,       4,  7}
1200 set(73)
1210 /*
1220 /*   AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2  WHISTLE2
1230 v={  14, 18, 13,  0, 12, 46,  0,  4,  3,  0,
1240      16, 15,  0,  3,  2,  1,  0,  8,  3,  0,
1250      16, 18,  0,  0,  7, 41,  0,  2,  7,  0, /* CON FBL
1260      14, 15,  0,  3,  2,  2,  0,  8,  7,  0,       4,  7}
1270 set(74)
1280 /*
1290 /*   AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2  SLAP BASS
1300 v={  31, 19,  6,  3,  4, 18,  3, 14,  3,  0,
1310      31,  9,  9,  3,  3, 44,  2,  1,  2,  0,
1320      31, 17,  0,  2,  2, 18,  0,  0,  0,  0, /* CON FBL
1330      31,  6,  0,  5,  1,  0,  2,  0,  7,  0,       3,  7}
1340 /*set(75)
1350 /*
1360 /*   AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2  HIHAT
1370 v={  31, 31,  0,  0,  1,  8,  0, 11,  0,  3,
1380      31, 28,  7,  0,  4, 17,  2, 12,  0,  3,
1390      31, 22,  0,  1,  4, 11,  1,  1,  0,  1, /* CON FBL
1400      31, 17,  7,  7,  1,  3,  2,  7,  0,  0,       1,  5}
1410 set(76)
1420 /*
1430 /*   AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2  CHORD2
1440 v={  18, 14,  0,  0,  1, 27,  0,  8,  7,  0,
1450      21, 11,  0,  6,  1,  0,  0,  4,  3,  0,
1460      21, 11,  0,  6,  1,  1,  0,  3,  7,  0, /* CON FBL
1470      21, 11,  0,  6,  1,  1,  0,  3,  3,  0,       6,  6}
1480 set(77)
1490 /*
1500 /*   AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2  CHORD3
1510 v={  18, 14,  0,  0,  1, 31,  0,  8,  7,  0,
1520      21, 11,  0,  6,  1,  3,  0,  8,  3,  0,
1530      31, 14,  0,  0,  1, 28,  0,  4,  3,  0, /* CON FBL
1540      21, 11,  0,  6,  1,  1,  0,  4,  7,  0,       4,  6}
1550 set(78)
1560 /*
1570 /*   AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2  SYNTH1
1580 v={  21, 11,  0, 10,  2,  9,  0,  4,  7,  0,
1590      19,  5,  0, 10,  3,  1,  0,  8,  3,  0,
1600      21, 11,  0, 10,  2,  8,  0,  3,  3,  0, /* CON FBL
1610      19,  5,  0, 10,  3,  1,  0,  6,  7,  0,       4,  0}
1620 set(79)
1630 /*
1640 /*   AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2  TOM
1650 v={  31,  0,  0,  0,  0,  0,  0, 15,  0,  0,
1660      31, 22, 15,  6,  5,  0,  0,  1,  0,  2,
1670      31, 16, 18,  8,  3,  0,  0,  2,  0,  0, /* CON FBL
1680      31, 16, 18,  8,  3,  0,  0,  1,  0,  0,       6,  7}
1690 set(80)
1700 /*
1710 /*   AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2  CHORD4
1720 v={  15,  5,  4,  0,  3, 22,  0,  8,  3,  0,
1730      21, 11,  0,  6,  1,  0,  1,  8,  7,  0,
1740      15,  5,  4,  0,  3, 26,  0,  4,  7,  0, /* CON FBL
1750      21, 11,  0,  5,  1,  0,  1,  4,  3,  0,       4,  5}
1760 set(81)
1770 /*
1780 /*   AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2  SYNTH2
1790 v={  26,  9,  7,  6,  4, 16,  0,  8,  1,  0,
1800      23,  0,  6, 11,  0,  0,  0,  8,  3,  0,
1810      26,  0,  4, 11,  0,  6,  0,  8,  7,  0, /* CON FBL
1820      26,  0,  4, 11,  0,  8,  0,  4,  4,  0,       6,  6}
1830 set(82)
1840 /*
1850 /*   AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2  CHORD5
1860 v={  31,  0,  4,  0,  0, 26,  0,  8,  3,  0,
1870      31,  0,  1,  9,  0,  5,  0,  4,  7,  0,
1880      31,  0,  4,  0,  0, 28,  0,  8,  7,  0, /* CON FBL
1890      31,  0,  1,  9,  0,  5,  0,  8,  3,  0,       4,  7}
1900 set(83)
1910 /*
1920 /*   AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2  CHORD6
1930 v={  19,  7,  3,  3,  3, 21,  0,  8,  3,  0,
1940      23, 14,  0,  5,  1,  1,  1,  8,  7,  0,
1950      19,  7,  3,  3,  3, 26,  0,  8,  7,  0, /* CON FBL
1960      23, 14,  0,  5,  1,  1,  1,  4,  3,  0,       4,  6}
1970 set(84)
1980 /*
1990 /*   AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2  NOISE1
2000 v={  31,  0,  0,  2,  0,  0,  0, 11,  3,  3,
2010      18, 18,  9,  8,  2,  0,  0, 14,  0,  3,
2020      31,  0,  0, 15,  0,  0,  0,  0,  0,  0, /* CON FBL
2030      31, 14, 14,  6,  3,  0,  0,  0,  0,  0,       4,  7}
2040 set(85)
2050 /*
2060 /*   AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2  SYNTH3
2070 v={  16, 11,  6,  1,  1, 17,  0,  4,  7,  0,
2080      16,  2,  1,  8,  1,  0,  1,  4,  3,  0,
2090      18,  5,  6,  2,  1, 29,  0,  4,  3,  0, /* CON FBL
2100      18,  2,  1,  8,  1, 16,  1,  4,  7,  0,       4,  6}
2110 set(86)
2120 /*
2130 /*   AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2  MAIN2
2140 v={  21, 15,  7,  7,  1,  8,  2,  4,  7,  0,
2150      21,  8,  0,  9,  2,  0,  1,  8,  7,  0,
2160      29, 15,  7,  7,  1, 10,  2,  2,  5,  0, /* CON FBL
2170      21,  8,  0,  9,  2,  0,  1,  4,  3,  0,       4,  0}
2180 set(87)
2190 /*
```


▶LIVE in'89のT-スクエアのBIG-CITYよかったです。思わずWAVE, TRUTH, SPORTSの3枚のアルバムを買ってしまいました。 山口 忠之 (17) 大阪府

```
2200 /*   AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2  MAIN3
2210 v={  21, 12,  6,  1,  1,  9,  2,  4,  7,  0,
2220      21,  7,  0,  5,  3,  1,  0,  8,  7,  0,
2230      29, 12,  6,  1,  1, 11,  2,  2,  5,  0, /* CON FBL
2240      21,  7,  0,  5,  3,  1,  0,  4,  3,  0,         4,  0}
2250 set(88)
2260 /*
2270 /*   AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2  MAIN4
2280 v={  25, 15,  0,  0,  1, 15,  0,  8,  3,  0,
2290      22,  0,  0,  6,  0,  5,  0,  4,  3,  0,
2300      22,  0,  0,  6,  0,  4,  0,  4,  3,  0, /* CON FBL
2310      29, 11,  3,  7,  2,  1,  0,  3,  3,  0,         5,  0}
2320 set(89)
2330 /*
2340 /*   AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2  CYM
2350 v={  31, 31,  0,  0,  1,  5,  0, 11,  0,  3,
2360      31, 28,  7,  0,  4, 18,  2, 12,  0,  3,
2370      31, 22,  0,  1,  4, 11,  1,  1,  0,  1, /* CON FBL
2380      31,  0,  5,  4,  0,  0,  1,  7,  0,  0,         1,  5}
2390 set(90)
2400 /*
2410 /*   AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2  CHORD7
2420 v={  31,  0,  0,  2,  0, 27,  0,  8,  5,  0,
2430      19, 15,  0,  4,  1,  2,  1,  4,  7,  0,
2440      31,  0,  0,  2,  0, 22,  0,  8,  4,  0, /* CON FBL
2450      19, 15,  0,  5,  1,  2,. 1,  8,  3,  0,         4,  6}
2460 set(91)
2470 /*
2480 /*   AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2  CHORD8
2490 v={  31,  0,  3,  0,  0, 26,  0,  8,  3,  0,
2500      31,  1,  0,  9,  1,  7,  0,  4,  3,  0,
2510      31,  0,  3,  0,  0, 32,  0, 14,  7,  0, /* CON FBL
2520      31,  1,  0,  9,  1,  7,  0,  8,  7,  0,         4,  5}
2530 set(92)
2540 /*
2550 /*   AR 1DR 2DR  RR 1DL  TL  RS MUL DT1 DT2
2560 v={   0,  0,  0, 15,  0,127,  0,  0,  0,  0,
2570       0,  0,  0, 15,  0,127,  0,  0,  0,  0,
2580       0,  0,  0, 15,  0,127,  0,  0,  0,  0, /* CON FBL
2590       0,  0,  0, 15,  0,127,  0,  0,  0,  0,         7,  0}
2600 /*set(71):set(89)
2610 /*
2620 endfunc
2630 /*
2640 /*  P L A Y   D A T A
2650 /*
2660 func MUS1()
2670 /*
2680 m_tempo(154)
2690 /*
2700 p(0)="[d.c.]r1@75@v127 l12 o3 p3 q8 y48,20 d-4dade-4e4 [co
da]q6
2710 p(1)="o2|:4f<cfcfc>:||:3a-<e-a-e-a-e->:|a-<a-e-a-e-e->
2720 p(2)="|:4b<b>b<e-eg->:||:4a<a>a<d-de>:|
2730 p(3)="q8o2f2q6|:4gg<g>:|<g6f4&f|:d-6d-:||:3>d-<d-:|e-6e-e-
>e-<e-e-6q8g4&g":p(3)=p(3)+p(3)
2740 p(4)="@80p1@v127l12|:3"+ls+ls+ll+":|
2750 o={0,1,1,2,2,3,4,30}
2760 t(1)
2770 /*
2780 /*
2790 p(0)="[d.c.]r1@75 v8   l12 o3 p1 q8 y49,48 d-4dade-4e4 @70
>[coda]@v119
2800 p(1)="|:f6q6f&f4q8f6f&f4&f1|1g-6g-&g-2.g-g-g-a-2.:||2g-6g-
&g-4&g-6rr4g-g-g-a-4|:3a-6:|
2810 p(2)="
2820 p(3)="@77v13|:b1&b1a1&a1:|
2830 p(4)="@81o2v13|:c2 d4ddd&d2d6c&c4 d-2d-2e-6>b-<|1g4e-2@81:
||2g4e-2&e-r6r2.
2840 o={0,1,2,3,4,30}
2850 t(2)
2860 /*
2870 /*
2880 p(0)="[d.c.]r1@75 v8   l12 o3 p2 q8 y50, 0 d-4dade-4e4 @70
>[coda]@v116
2890 p(1)="|:b-6q6b-&b-4q8b-6a&a4&a1|1b-6b-&b-2.b-b-b-<c2.>:||2
b-6b-&b-4&b-rrr4b-b-b-<c4c6c6c6
2900 p(3)="@78v13|:e-1&e-1d-1&d-1:|
2910 p(4)="@82o2v13|:f2 g4ggg&g2g6f&f4 f2f2gre-|1b-4g2@81:||2b-
4g2&gr6r2.
2920 t(3)
2930 /*
2940 /*
2950 p(0)="[d.c.]r1y51,60@90q1@v12l14o4cfcb       @70 v13  l12 o
3 p3 q8 y51,16[coda]@v118
2960 p(1)="|:p1e-6q6d&d4q8d6c&c4&p3c1p1 |1d-6d-&d-2.d-d-d-e-2.:
||2d-6d-&d-4r2d-d-d-e-4|:3e-6:|
2970 p(3)="@79p1|:4v12b6bv9bv8bv7b:||:4v12a6av9av8av7a:|":p(3)=
p(3)+p(3)
2980 p(4)="@81o2v13p1|:a2 brrb<c>b&b2 b6a&a4 q6a-2<q8d-2>b-6g|1
<e-4>b-2:||2<e-4>b-2&b-r6r2.
2990 t(4)
3000 /*
3010 /*
3020 p(0)="[d.c.]r1y52,00@90q1@v123l4o4p2cp1fp2cb@70 v13  l12 o
3 p3 q8 y52,28[coda]@v118
3030 p(1)="p2e-6q6d&d4q8d6c&c4@72l24q1>|:v15fv5f<v15cv11cv15fv1
1fv15cv11cv15fv11fv15cv11c>:|<q8@70l12v12 p2d-6d-&d-2.d-d-d-e-2.
3040 p(2)="p2e-6q6d&d4q8d6c&c4@72l24q1>|:v15fv5f<v15cv11cv15fv1
1fv15cv11cv15fv11fv15cv11c>:|<q8@70l12v12 p2d-6d-&d-4r2d-d-d-e-4
|:3e-6:|
3050 p(3)="@79p2|:4v12b6bv9bv8bv7b:||:4v12a6av9av8av7a:|":p(3)=
p(3)+p(3)
3060 p(4)="@81o2v13p2|:a2 brrb<c>b&b2 b6a&a4 q6a-2<q8d-2>b-6g|1
<e-4>b-2:||2<e-4>b-2&b-r6r2.
3070 t(5)
3080 /*
3090 /*
3100 p(0)="[d.c.]r1r1    @71 v12  l12 o3 p3 q8 y53,16   [coda]
3110 p(1)="v11c&v9fv12b-<e-6&@l4|:4y53,16e-&y53,220d&y53,16e-&y
53,68e-&:|y53,16y8,5l12dc>b-@l3
3120 pol(16,-34,16,6,10):p(2)=l+"&
3130 p(3)="@l4|:6y53,16c&y53,144>b&y53,16<c&y53,128c&:|y53,16l1
2>g-b-g-<d-6&@l4|:4y53,16d-&y53,200c&y53,16d-&y53,88d-&:|y53,16y
8,5>
3140 pol(16,-24,12,6, 5):p(4)=l+"&y53,16|:3y53,16g-&y53,220f&y5
3,16g-&y53,68g-&:|y53,16y8,5l12
3150 p(5)="<fe-d-c>b-a-<c>b-a-<@l4|:3y53,16a-&y53,240g&y53,16a-
&y53,48a-&:|y8,5l12>
3160 p(6)="<c4&c12&@l4
3170 pol(16,80,20,6, 0):p(7)=l+"y53,16l12g-b-g-@l3&
3180 pol(16,-44,16,6,10):p(8)=l+"&@l4|:3y53,16d-&y53,220c&y53,1
6d-&y53,68d-&:|y8,5
3190 p(9)="g-12&|:5y53,16g-&y53,144f&y53,16g-&y53,128g-&:|y8,5y
53,16
3200 p(10)="l12>a-ga-b-a-b-<c>b-<cd
3210 p(11)="cd@73v15o3y53,16 @l3
3220 pol(16,-72,8,6,1):p(12)=l+"&y53,16e-8r6>b12
3230 p(13)="@l4g-4&|:12y53,16g-&y53,148f&y53,16g-&y53,140g-&:|y
8,5y53,16 l12g-b<e-d-6
3240 p(14)="&d-a4@l4e4&|:15y53,16e&y53,148e-&y53,16e&y53,140e&:
|y53,16y8,5@l3
3250 p(15)="l12e-4&e-r>b
3260 p(16)=">a<a4@l4&|:16y53,16a&y53,148a-&y53,16a&y53,140a&:|y
53,16y8,5l12rr
3270 p(17)="@82q8v13o2p2l12y53, 8
3280 p(18)="a2b&d&gb<c>b2rb6a&a4a-2<d-2>b-rg<e-2.
3290 p(19)="@71v12o3q8@l4g&a-&a8&
3300 pol(16,68,16,6,9):p(20)=l+"y53,16
3310 p(21)="l12bdbb&<c&>b&b2&bra&a4&a-2 <d-r>a-rfrb-rg<e-4>b-rg
re-r
3320 p(22)="@80p2@v127l12|:3"+ls+ls+ll+":|
3330 o={0,1,2,3,4,5,1,6,7,8,9,10,11,12,13,14,15,13,16,17,18,19,
20,21,22,30}
3340 t(6)
3350 /*
3360 /*
3370 p(0)="[d.c.]r1r1@l1r@71 v11  l12 o3 p3 q8 y54,36 r6[coda]
3380 p(1)="v10c&v8fv11b-<e-2dc>b-@l3
3390 pol(36,-34,16,7,10):p(2)=l+"&
3400 p(3)="c3&c8&c24l12>g-b-g-<d-2>@l4
3410 pol(36,-24,12,7, 5):p(4)=l+"&y54,36g-4l12
3420 p(5)="<fe-d-c>b-a-<c>b-a-<a-4>
3430 pol(36,76,20,7, 0):p(7)=l+"y54,36l12g-b-g-@l3&
3440 pol(36,-44,16,7,10):p(8)=l+"&y54,36d-4&
3450 p(9)="g-2
3460 p(11)="rr@73v13o3y54,40 @l3
3470 pol(40,-72,8,7,1):p(12)=l+"&y54,40e-8r6>l12b
3480 p(13)="g-4&g-1g-b<e-d-6
3490 p(14)="&d-a4e1&e2@l3
3500 p(16)=">a<a4&a1&a6rr
3510 p(17)="@82q8v13o2p1l12y54,36
3520 p(19)="@71v11o3q8@l4r6a6
3530 pol(40,68,16,7,9):p(20)=l+"y54,40
3540 p(22)="r2.r12
3550 t(7)
3560 /*
3570 /*
3580 p(0)="[d.c.]@76@v127 l24 o4 p3 q1 y55,20 y15,0 y3,3 y2,14c
y2,15r|:y2,15c12:|y2,13q8cq1y2,13r|:y2,13c12:|y2,14c12|:y2,15cc:
|q8|:y2,15c:|y2,15rrq1y2,14cc q8l12y2,5c|:q3y2,15c:||:q8y2,5cy2,
16q3cy2,16c:|q8y2,5c|:y2,14p2c:|q1p3[coda]
3590 p(1)="y2,23ccy2,23cy2,14q8cq1cy2,23cy2,23ccy2,23cy2,14q8cr
q1c
3600 p(2)="|:y2, 5ccy2,23cy2,14q8cq1cy2,23cy2,23ccy2,23cy2,14q8
crq1c
3610 p(3)="y2,23ccy2,23cy2,14q8cq1cy2,23cy2,23ccy2,23cy2,14q8cr
q1c
3620 p(4)="|1y2,23ccy2,23cy2,14q8cq1cy2,23cy2,23cy2,14cy2,14cy2
,14q8cy2,14ry2,14q1c:|
3630 p(5)="|2y2,23ccy2,23cy2,14q8cq1cy2,15cy2,23cy2,14cy2,15cy2
,14p1q8cy2,14c&y2,14q1cp3
3640 p(6)="|:y2,23p1q8c&y2,23c&y2,23cq1p3cy2,23cy2,23c ccy2,23c
y2,14ccc
3650 p(7)="y2,23ccy2,23cy2,14ccp2c24c24p3 y2,23ccy2,23cy2,14ccc
3660 p(8)="y2,23ccy2,23cy2,14ccp2c24c24p3 y2,23cy2,23cy2,23cy2,
14ccc
3670 p(9)="|1y2,23ccy2,23cy2,14ccc y2,23cy2,14cy2,23ccy2,14cc:|
3680 p(10)="|2y2,23ccy2,23cy2,14ccc y2,23cy2,14cy2,14cy2,23cy2,
14cy2,14c
3690 p(11)="y2,5rcrq8cq1cr c@80"+hl+ml+"@76cy2,23q8c&cy2,23rq1y
2,14ccy2,23c y2,23ccy2,23cy2,14ccc
3700 p(12)="q1|:y2,23c&cy2,23ry2,14ccy2,23c|1y2,23c&cy2,23cy2,1
4cy2,23cc:||2y2,14cy2,15cy2,15@80"+hs+"@76y2,14cy2,15cy2,15@80"+
ms+"@76
3710 p(13)="y2,5cq8cq1cq8cq1cr r@80"+hl+ml+"@76cy2,23q8c&cy2,23
cq1y2,14ccy2,23c p3|:q5y2,14cy2,15q1cy2,15c:|
3720 p(14)="|:y2,23c&cy2,23ry2,14ccy2,23c|1y2,13c&cy2,13rc&y2,1
3cr:||2y2,16c&y2,16cy2,15@80"+hs+"@76y2,14c&y2,14cy2,14@80"+ms
3730 p(15)="@90p2t148y2,5crrr @76q1ccq8cq1cy2,16cy2,16q8cy2,15q
1cy2,14c
3740 o={0,2,3,1,4,5,6,7,8,9,10,11,12,13,14,15,30}
3750 t(8)
3760 /*
3770 /*
3780 endfunc
3790 /*
3800 func MUS2()
3810 /*
3820 p(0)="@75@v127 l12 o3 p3 q6 y48,24 t154
3830 p(1)="|:8ff>f<:||:6e-e->e-<:|e-e-e->e-e-<e-|:8d-d->d-<:||:
7e-e->e-<:|
3840 p(2)=p(1)+"e-e->e-<
3850 p(3)=p(1)+"e->e-e-<
3860 o={0,2,2,2,3,30}
3870 t(1)
3880 /*
3890 /*
3900 p(0)="@84 v12  l 3 o2 p3 q6 y49,24
3910 p(1)="q6|:4c6c:||:c6ce-6e-:||:4d-6d-:||:e-6e-3:|q8d-4e-4{e
-rf&}4f4
3920 p(2)="q6|:4a6a:||:4b-6b-:||:4a-6a-:||:b-6b-:|q8b-4b-4{b-r<
```

▶LIVE in '89に「うけねらい」のプログラムを投稿しようとしてるため，学校のテストなんかやってられっかぁ。
椎野　篤哉（20）神奈川県

```
c&}4c4>
 3930 p(3)="v1312q8crcre-re-rd-rd-re-rv12d-4e-4{e-rf&}4f413
 3940 o={0,1,2,3,2,30}
 3950 t(2)
 3960 /*
 3970 /*
 3980 p(0)="@84 v12  1 3 o2 p3 q6 y50,20
 3990 p(1)="q6|:4f6f:||:4g6g:||:4f6f:||:g6g:|q8f4g4{grp2g&}4g4p3
 4000 p(2)=p(1)
 4010 p(3)="v1312q8frfrgrgrfrfrgrv12f4g4{grp2g&}4g4p313
 4020 t(3)
 4030 /*
 4040 /*
 4050 p(0)="@84 v12  1 3 o2 p1 q6 y51,16
 4060 p(1)="o2q6|:4a6a:||:4b-6b-:||:4a-6a-:||:b-6b-:|q8b-4b-4{b-
r<c&}4c4>
 4070 p(2)="@86q8o3112p2v11|:7ggraab-:|ggraa<c>|:6ggra-a-b-:|b-4
b-4b-r<c&c4@84v1212o2p1q6>
 4080 p(3)="v1312o2q8ararb-rb-ra-ra-rb-rv12b-4b-4{b-r<c&}4c4>13
 4090 t(4)
 4100 /*
 4110 /*
 4120 p(0)="@84 v12  1 3 o2 p2 q6 y52,28
 4130 p(2)="@86q8o3112p1v11|:4ccrccc:||:ccrccce-e-re-e-e-:||:5d-
d-rd-d-d-:|<d-d-rd-d-d->d-4e-4frf&f4@84v1212o2p2q6
 4140 t(5)
 4150 /*
 4160 /*
 4170 p(0)="@71 v12  112 o3 p3 q8 y53,16   |:4v11f&c&f&v12gab-&:
|y8,5
 4180 p(1)="b-agagfgfe-fe-d e-b-e-<b-e->b-|:<e-6>b-:|
 4190 p(2)="y54,80<d-ce-a-ge-ff+fe-d-e->a-<y54,44d-e-a-e-d->a-<e
-&d-cd-r e-de-d-4cd-c>b-a-b-<
 4200 p(3)="@14e-6&|:4y53,16e-&y53,144d&y53,16e-&y53,128e-&:|y53
,16y8,5<e-4&e-12&@12v13
 4210 pol(16,68,12,6,3):p(4)=l+"y8,5y53,16r24
 4220 p(5)="@87@v123112o4q8
 4230 p(6)="c2>b-2a6b-r4@88<c>b-afe-d|:6e-de-:|fe-d>b-a-f e-fe-f
gfga-gb-a-b-|:<e->b-g:|<e-fe-d->a-g
 4240 p(7)="|:v14g&a-&g&a-&g&a-:|<v14e-1
 4250 p(8)="@89v12124o4q8y53,16
 4260 p(9)="c>&b-&a&f&a&g&f&e-&c&d&e-&f& e-&d&e-&d&c&b-&d&c&>b-&
a-&g&f& e-&d&c&>b-&a-&g&f&e-&d&c&>b-&a-& g&a-&b-&<c&d&e-&f&g&a-&
b-&<c&d
 4270 p(10)="<|:5b-&g&e-&:|>b-&<d&>e-<&f&g&f&e-&d&c< e-&>b-&f<d&
>b-&f<e-&>b-&f<d&>b-&f<e-&>b-&f<f&>b-&f<e-&>b-&f<d&>b-&f
 4280 p(11)="<|:4a-&f&e-&a-&f&e-&>a-&<e-&f>a-&<a-&f:|
 4290 p(11)="<|:4a-&f&e-&f&e-&a-&e-&>a-&<f&>a-&<f&a-:|
 4300 p(12)=">>a-&a&b&<c&c+&d> b-&<c&c+&d&d+&e c&c+&d&d+&e&f e-&
e
 4310 p(13)="&f&f+&g&g+ b-4b-4b-12r12b-&a& a-&g&g-&f&e&e-
 4320 p(14)="c6b-arb->gfe-agf|:3e-g<e->:|e-g<d>|:e-g<c>:|e-gb-e-
g<c>
 4330 p(15)="v13e-e-dv12e-e-dv14
 4340 p(16)="fe-fb-4e-6d-r4|:e-d-e-a-4|1e-6d-r4:||2fgfga-ga-b-a-
g
 4350 p(17)="a-gv13f4g4grg&g4
 4360 o={0,1,2,3,4,5,6,7,8,9,10,11,12,13,5,14,15,16,17,30}
 4370 t(6)
 4380 /*
 4390 /*
 4400 p(0)="@71 v11  112 o3 p3 q8 y54,44 r6|:4v10f&c&f&v11gab-&:
|y8,6
 4410 p(3)="e-2<e-4&e-@14
 4420 p(4)="
 4430 p(5)="@87@v117112o4q8y54,44r6
 4440 p(7)="|:ga-&g&a-&g&a-:|<e-2&e-6rr
 4450 p(8)="@89v11124o4q8y54,36r6
 4460 p(13)="f4g4g12r12g&g-& f&e&e-&d&d-&c
 4470 p(15)="v12e-e-dv11e-e-dv13
 4480 p(17)="v13b-4b-4b-r<c&c4
 4490 t(7)
 4500 /*
 4510 /*
 4520 p(0)="@76@v127 112 o4 p3 q1 y55,20 y15,0 y3,3
 4530 p(1)="y2,23q7c&c&y2,23cy2,14c&c&y2,23c
 4540 p(2)="y2,23q8cq1cy2,23cy2,14q8cq1cc
 4550 p(3)="y2,23q8cq1cy2,23cy2,14q8cy2,23q1cc
 4560 p(4)="y2,23q8cq1cy2,23cy2,16q8cy2,15q1cc
 4570 p(5)="y2,23c&c@90p2y2,5cr4@76y15,0p3
 4580 p(6)="y2,23q7c&c&cy2,14c&c&y2,23c
 4590 p(7)="@80|:124"+hs+"y8,7"+hs+"112"+ms+ls+":|@90
 4600 p(8)="y2,23p1cry2,23ry2,14rry2,23r y2,23p3ery2,23ry2,14ry2
,16rr
 4610 p(9)="y2,23p2ary2,23ry2,14rry2,23r y2,23p3ery2,23ry2,14ry2
,16rr
 4620 p(10)="y2,23p2ary2,23ry2,14ry2,16rr y2,23p3ery2,23ry2,14ry
2,16rr
 4630 p(11)="y2,23p2ary2,23ry2,14rry2,23r p1dy2,23ry2,23ry2,14ry
2,16rr
 4640 p(12)="y2,23p2gy2,16ry2,23ry2,15p1cy2,15ry2,23r y2,14rry2,
5p3cr4@76
 4650 o={0,1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,5,
 4660       6,2,1,3,1,2,1,4,1,2,1,3,1,2,1,7,
 4670       8,9,10,11,8,9,10,12,
 4680       1,2,1,3,1,2,1,4,1,2,1,3,1,2,1,5,30}
 4690 t(8)
 4700 /*
 4710 /*
 4720 endfunc
 4730 /*
 4740 func MUS3()
 4750 /*
 4760 p(0)="@75@v127 112 o2 p3 q6 y48,24 t154
 4770 p(1)="|:4a-6a-a-a-a-:||:4b6bbbb:|<|:4d-6d-d-d-d-:|v15
 4780 l="e-dd-c>bb-aa-gg-fe":l=l+l:p(2)=l+"o2@v127
 4790 p(3)="@v127>f4a-4<e-4a-4>e-e-<e->e-<e->e-a-4q8d-4&d-1&d-1
 4800 p(4)="@75o3
 4810 p(30)="[*]
 4820 o={0,1,2,1,3,4,30}
 4830 t(1)
 4840 /*
 4850 /*
 4860 p(0)="@91 v12  112 o2 p3 q6 y49,24
 4870 p(1)="p3v12crce-2.crce-4e-2|:e-re-g-2.:|frfa-2.frfa-4a-2p1
v10<
 4880 p(2)="v10@84p1>"+l+"@91v12o2
 4890 p(3)="@83q8y49,16p3o1v14ffv9fv14b-b-v9b-v14<e-e-v9e-v14a-a
-v9a-v14> b-b-v9b-v14<e-e-v9e-v14a-a-v9a-<v14d-4&d-1&d-1
 4900 p(4)="@70v13o2q8y49,48
 4910 o={0,1,2,1,3,4,30}
 4920 t(2)
 4930 /*
 4940 /*
 4950 p(0)="@91 v12  112 o2 p3 q6 y50,12
 4960 p(1)="p3v12e-re-a-2.e-re-a-4a-2g-rg-b2.g-rg-b2b-4a-ra-<d-2
.>a-ra-<d-4d-2p2v10
 4970 p(2)="v10@84p2>"+l+"@91v12o2
 4980 p(3)="@83q8y50,24p3o1v14b-b-v9b-<v14e-e-v9e-v14a-a-v9a-v14
<d-d-v9d-> v14e-e-v9e-v14a-a-v9a-v14<d-d-v9d-v14g-4&g-1&g-1
 4990 p(4)="@70v13o2q8y50,00
 5000 t(3)
 5010 /*
 5020 /*
 5030 p(0)="@91 v12  112 o2 p1 q6 y51,16
 5040 p(1)="a-ra-<c2.>a-ra-<c4c2>brb<e-2.>brb<e-4d-2d-rd-f2.d-rd
-f4f2
 5050 p(2)="@75v12p1o3"+l+"@91v12p1o2
 5060 p(3)="@83q8y51, 0p1o2v14e-e-v9e-v14a-a-v9a-<v14d-d-v9d-v14
g-g-v9g-> v14a-a-v9a-<v14d-d-v9d-v14g-g-v9g-p3v14b4&b1&b1
 5070 p(4)="@70v13o3q8y51,16
 5080 t(4)
 5090 /*
 5100 /*
 5110 p(0)="@91 v12  112 o2 p2 q6 y52,32
 5120 p(2)="@75v12p2o3"+l+"@91v12p2o2
 5130 p(3)="@83q8y52,28p2o2v14e-e-v9e-v14a-a-v9a-<v14d-d-v9d-v14
g-g-v9g-> v14a-a-v9a-<v14d-d-v9d-v14g-g-v9g-p3v14b4&b1&b1
 5140 p(4)="@70v13o3q8y52,28
 5150 t(5)
 5160 /*
 5170 /*
 5180 p(0)="@86 v12  112 o2 p2 q8 y53,16 a-rr<c4<@71
 5190 p(1)="e-4d-4cd-ce-d-c>
 5200 p(2)="@14a-4&|:9y53,16a-&y53,180g&y53,16a-&y53,92a-&:|y53,
16y8,5112
 5210 p(3)="<fe-fg-fg-b-ra-rg-fe-
 5220 p(4)="@14d-6&|:9y53,16d-&y53,180c&y53,16d-&y53,92d-&:|y53,
16y8,5112
 5230 p(5)="|:5fg-:|<d-6
 5240 p(6)="@14f6&|:4y53,16f&y53,180f-&y53,16f&y53,92f&:|y53,16y
8,5112>v9o3"+l+"v12o4
 5250 p(7)="@14e-4&|:3y53,16e-&y53,200d&y53,16e-&y53,72e-&:|y53,
16y8,5112d-c>b-<c>b-
 5260 p(8)="e-&@14e-4&|:9y53,16e-&y53,180d&y53,16e-&y53,92e-&:|y
53,16y8,5112
 5270 p(9)="g-b-g-<e-4>g-b-g-a-6g-fe-d->
 5280 p(10)="@14b4&|:7y53,16b&y53,180b-&y53,16b&y53,92b&:|y53,16
y8,5112
 5290 p(11)="r4r6<|:fg-<d->:|<e-6d-cd-e-a-6e-6d-6>
 5300 p(12)="v12e-4a-&e-&a-<d-6&@14|:4y53,16d-&y53,220c&y53,16d-
&y53,68d-&:|112v11>|:3c>b-<:|
 5310 p(13)="e-4@92p1v13@16|:18y53,16g-&y53,212f&y53,16g-&y53,60
g-&:|y53,16y8,5
 5320 p(14)="@71v12112o3q8
 5330 o={0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,30}
 5340 t(6)
 5350 /*
 5360 /*
 5370 p(0)="@86 v12  112 o2 p1 q8 y54,40 a-rr<c4<@71v11r6
 5380 p(2)="a-1
 5390 p(4)="d-2.&d-6
 5400 p(6)="f4&frv8o3"+l+"v11o4r
 5410 p(7)="e-2d-c>b-<c>b-
 5420 p(8)="e-&e-1
 5430 p(10)="b2.&b12
 5440 p(12)="v10e-4a-&e-&a-<d-2>|:3c>b-<:|
 5450 p(13)="e- @92p2v13@16|:18y54,40d-&y54,236c&y54,40d-&y54,84
d-&:|y54,40y8,6
 5460 p(14)="@71v11112o3q8 r6
 5470 t(7)
 5480 /*
 5490 /*
 5500 p(0)="@90@v127 112 o4 p3 q1 y55,20 y15,0 y3,3
 5510 p(1)="y2,23p2cry2,23ry2,14cry2,23r y2,23rry2,23ry2,14rrr
 5520 p(2)="y2,23rry2,23rp1y2,14gy2,14ry2,15r y2,23rry2,23ry2,14
p2crr@v0
 5530 p(3)="y2,23rry2,23ry2,14@v127p1gry2,23r y2,23r@76q1cy2,23r
y2,14q6c4@90
 5540 p(4)="y2,13rry2,13r@76q8c&y2,14c&y2,15c y2,13rq1cq6y2,13ry
2,14c4@90
 5550 p(5)="y2,23rry2,23ry2,14p1gry2,23r y2,23r@76q1cy2,23ry2,14
q6c4@90
 5560 p(6)="y2,23rry2,23ry2,14p1gy2,14ry2,15r y2,23rry2,23ry2,14
p2crr@v0@76q1
 5570 p(7)="y2,23r@v127ccy2,14rcy2,23@80"+ms+"@76y2,23rcy2,23@80
"+ls+"@76y2,14rcy2,23c y2,23rccy2,14rcy2,23c
 5580 p(8)="y2,23ry2,16cy2,15cy2,14r|:y2,14c:|@90
 5590 p(9)="@v127y2,23p1crr@80q1"+hs+ms+"y2,23"+ls+"y2,14r@76|:5
y2,15c:|@90y2,23p1gry2,23rp2cy2,13ry2,23r
 5600 p(10)="@90y2,16p1gy2,14ry2,15ry2,5p2crr|:4y2,23r4:| y2,23r
y2,14ry2,15r|:3y2,14ry2,15ry2,15r:|@76q1
 5610 o={0,1,2,3,4,5,6,7,8,1,2,3,4,5,6,9,10,30}
 5620 t(8)
 5630 /*
 5640 /*
 5650 endfunc
 5660 func trfree()
 5670 for i=1 to 8
 5680  print "TRK";i;" ---";m_free(i)
 5690 next
 5700 endfunc
```

X68000用
カードゲーム

ばばぬき

Mounai Toshiyuki
毛内 俊行

X68000で動くゲーム「ばばぬき」です。言わずと知れたトランプの代表的なゲームで，プログラムはX-BASICで書かれています。皆さんも，手ごろなトランプゲームのパソコン版を作ってみてはどうでしょうか？

気軽に遊べるゲームがほしい

ある日曜の昼下がり，ひとりの少年が部屋の片隅でまんが本に埋もれているX68000の前に座った。彼はおもむろにディスクのたくさん入った箱をのぞき込んでつぶやいた。

「暇だからゲームでもしようかなと思ったんだけど，いったいなにで遊ぼうかな？ そろそろR-TYPEは飽きたし，ジェノサイドは疲れるし……たまにはなにも考えずに，のほほーんとしながら遊べるゲームをしてみたいもんだなー」。彼はしばらく，いろいろなゲームディスクを引っ張り出していたが，やがて1枚のディスクを手にして叫んだ。

「そうだ，こんなときはこれに限る!」。そういって彼が取り出したのは，銀のラベルの貼られたアフターバーナーのディスクでも，緑と黄色のカラフルな色彩の，サンダーフォースIIのディスクでもなかった。手書きのラベルに，サインペンで大きく「ばばぬき」と書かれた，X-BASIC用のプログラムの入った地味なディスクだった。

*　　　*　　　*

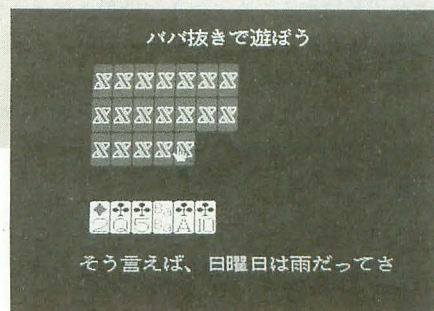
みなさんも，ここに登場した少年のように，「ゲームはやりたいんだけどなぁ……」というような状況を味わったことがあると思います。近頃はX68000にも結構たくさんのゲームが出揃い，退屈はしないのですが，どうもこれらのゲームというのはやたら頭を使ったり，反射神経が鈍いとできなかったりというものが多く，1ゲームもプレイすると，もう疲れてしまいます。

「たまにはほっと一息つける，マイルドセブンのようなゲームがほしい！」そう思った私は，マウス一丁で操作できて，そのうえいざとなればテレビを見ながらでもできるゲームを作ろうと思い立ったわけです。今回は，誰もが知ってて退屈せず，それでいて手軽なゲームというわけで，トランプを使ったもっともポピュラーなゲーム，「ばばぬき」を紹介しようと思います。

プログラムの入力と実行

このゲームはすべてX-BASICで書かれています。プログラムは全部で2本に分かれており，リスト1がトランプのパターンの定義，リスト2がゲームプログラム本体です。それぞれ，BASICから入力してセーブしてください。

実行するときは，まずリスト1を実行してカードパターンを定義してください。忘れるとカードがグシャグシャになります。リスト1ではスプライトパターンにカードを定義するので，1回実行してしまえば，あとは自分でスプライトをいじったりグラディウスをやったり電源を切ったりしない限り，二度と実行する必要はありません。

カードのパターンは同じようなパターンを赤と黒の2種類定義しなければなりません。リスト1はプログラムを短くする代わりに，同じデータを赤と黒の1回ずつ処理して定義するので，実行が終了するまで30秒以上かかります。それが嫌な人は一度リスト1を実行したあと，システムディスクに入ってるDEFSPTOOLでパターン作成用のプログラムを新しく作ってしまいましょう。次回からはこの新しく作成したプログラムを使えば，一瞬にしてパターンを作成してくれます。ただし，黒の色はSP_COLORを使ってパレットコード1にカラーコード1を定義しているので注意してください。

カードの定義が終了したら，あとはゲームを実行するだけです。プレイヤーは，あなたを含めて4人いますが，あなたのほかはみんなコンピュータです。人間のプレイヤーは2人以上にはなりません（よく考えれば当たり前でしょう)。リスト2を実行するとゲームが始まります。RUNすると，自動的にシャッフルをして，カードを配り，各プレイヤーのカードを整理してくれます。それが終わると，画面に手の形をしたマウスカーソルが表示されて，あなたの番になります。カードの引き方ですが，マウスを左右に動かして任意のカードにカーソルを合わせ，マウスの左ボタンをクリックすれば，そのカードを引くことができます。引いたカードを捨てることができない場合，そのカードは手札のいちばん右に添えられますが，プレイヤーがコンピュータの場合はその後，手札をシャッフルするのでカードの位置は1回引くたびに変わります。

カードを引いたら，あとは自分の番が回ってくるまで，コンピュータのプレイの見物でもしていてください。プレイヤーはあなたのほかに3人います。知ってるとは思いますが，最後までジョーカーを持っていた人が負けです（もっとも，手札にジョーカーが残っているほうが最後までスリルがあって面白いかもしれません)。もし，自分がほかのプレイヤーより先にあがってしまったら，ゲーム終了まで，コンピュータ同士のプレイが続けられます。

ゲームをプレイ中，画面の下にいろいろなメッセージが表示され，とても賑やかです。これらのメッセージは，各プレイヤーがカードを引くときに表示されるようになっていますが，しばらくカードを引かないでいると各プレイヤーが勝手にいろんな話を始めます。これらのメッセージは配列

変数に定義されていて，ランダムに選ばれて表示されますがまったくの乱数ではなく，ある程度引いたカードなどによってリアクションが異なります。

また，これはおまけですが，リスト2の100行目の変数定義で，demo＝1にすると人間はプレイできなくなり，4人ともコンピュータがプレイするようになります。ぼーっと眺めているのも面白いかもしれません。

それから150行目を，screen0,0,0,1に書き換えて，スーパーインポーズでテレビを見ながらゲームをすることもできます。ときにはテレビを見ながらゲームをしたりするのも気分がいいものですし，見たいテレビ番組がナイター中継などで時間が押されている時などでも，ゲームで遊びながら中継が終わるのを待つことができます（なんて便利なんだろう)。もっともこのときは，メッセージが画面の外にはみ出してしまいますので注意が必要です。

画面上のカード

今回のばばぬきで用いたカードパターンは16×16ドットのスプライトを縦に2つ並べて16×32ドットで表現しています。これだけのドット数では，さすがにリアルなカード描写はできないので，カードの内容を思い切って簡略化しました。カードの上半分はスペード，ハートなどのスート模様を表現して，下半分で数を表しています。

本当はグラフィックを用いてもっとリアルなカードを作ろうとしたのですが，絵心のない私にとって，クイーンやキングの絵を書くのは死ぬほど大変だったので，ここらへんはとっても手抜きになってしまいました。このカードの文字や絵柄が気に入らない人は，リスト1を自分の好きなように書き換えてください。もっとも，カードのデザインは根本的に決まっていますので，カード全体に絵を描いたり，カードの上下を入れ替えたりというようなことはできません。

一応，このパターンさえあればトランプと名のつくものすべてに対応することができますが，個人的に少し手抜きをしすぎたかなと思います。誰かカードゲームを作ろうと思い立った人がいたなら，今度はグラフィックを使ってもっとかっこいいカードを作ってほしいものです（期待してます)。

もっとカードゲームを

余談ですが私は初め，神経衰弱を作ろうと思い，このパターンを作っていたのですが，作っている途中，勢い余ってジョーカーのパターンまで作ってしまったので，どうせならばばぬきにしようと思ったわけです。

さらに余談ですが，このジョーカーというカードを使ったゲームは，外国には一切ないそうです。外国にもばばぬきによく似たゲームで「ホールドメイド」というのがあるのですが，これはジョーカーを使う代わりに，クイーンを1枚抜き取って行う，いわゆる我々のいう「じじぬき」によく似たゲームになっています。そのほか，ジョーカーを使うゲームというのは日本で生まれたものか，さもなくば日本にきてジョーカーを使うルールに変えられたものなんだそうです（知らなかったでしょ)。と，これはまったくの余談でしたね。失礼しました。

さて，先ほども触れたように今回は「マイルドセブンのようなゲーム」としてばばぬきを紹介しましたが，ほかにも「キャスター」や「PMスーパーライト」，なかには「ピース」が好きだとおっしゃる方，さらには「パイポ」を使ってるなんていう方もいることと思います。そう，ばばぬきのほかにも気軽に遊べるゲームというのは，けっこうたくさんあるはずです。トランプを使ったゲームとしては「大富豪」(大貧民）や，「神経衰弱」なんかは簡単にできると思いますし，別にカードゲームにこだわらなくても，単純で面白いゲームというのはたくさんあるはずです。

実はこういうゲームを誰かが作ってきてくれると，とっても嬉しいなあと私は思うんですけどね。そりゃ私が作れば，作れないことはないんですけど，Oh!Xって読者のための雑誌なんだから，スタッフの私より読者のみんなが参加してくれなくちゃいけないんですよね。うーんほしいなぁ，大富豪に神経衰弱。……できれば神経衰弱なんか，カードを画面上にぱぁーっとばらまいた感じの，ちょっとくらいカードが重なったり，斜めに置かれたカードがあったりして……。ちょっと難しいかな？

とにかく，極端に面白いとかいうわけではないのだけれど，遊んでいるとなんとなくほっとするようなゲームって，たくさんほしいですよね。この手のゲームって，市販のゲームと比べるとインパクトが弱いけど，ピコピコゲームよりちょっとクオリティが高いという，ちょうど中途半端な位置づけになるような気がします。いったいこの手のゲームって，どう呼んでやったらいいのでしょう？　誰か「ピコピコゲーム」のような，ズバリとはまった名称を考えてはくれないかな？　そしてみんなで新しいジャンルのゲームのパイオニアになろうじゃありませんか！

リスト1　カード設定

```
 10 int i,j,k
 20 dim char c1(255),c2(255),c3(255),c4(255),c5(255)
 30 dim char c6(255),c7(255),c8(255),c9(255),ct(255)
 40 dim char cj(255),cq(255),ck(255)
 50 /*
 60 screen 1,2,1,1:cls:console ,,0
 70 locate 20,15:print"しばらくお待ち下さい  ☆☆☆☆"
 80 locate 42,15
 90 sp_init()
100 dim char hi(255)={
110      0,15,15,15,15,15,15,15,15,15,15,15,15,15, 0, 0,
120     15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
130     15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
140     15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
150     15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
160     15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
170     15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
180     15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
190     15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
200     15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
210     15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
220     15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
230     15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
240     15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
250     15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
260     15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0
270 }
280 dim char lw(255)={
290     15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
300     15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
310     15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
320     15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
330     15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
340     15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
350     15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
360     15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
```

```
370      15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
380      15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
390      15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
400      15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
410      15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
420      15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
430      15,15,15,15,15,15,15,15,15,15,15,15,15,15,15, 0,
440       0,15,15,15,15,15,15,15,15,15,15,15,15,15, 0, 0
450 }
460 dim char spd(255)={
470      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
480      0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,
490      0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,
500      0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,
510      0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,
520      0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,0,
530      0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,0,
540      0,0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,
550      0,0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,
560      0,0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,
570      0,0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,
580      0,0,0,1,1,1,0,1,0,1,1,1,0,0,0,0,
590      0,0,0,1,1,0,0,1,0,0,1,1,0,0,0,0,
600      0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,
610      0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,
620      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
630 }
640 dim char hrt(255)={
650      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
660      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
670      0,0,0,0,1,1,0,0,0,1,1,0,0,0,0,0,
680      0,0,0,1,1,1,1,0,1,1,1,1,0,0,0,0,
690      0,0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,
700      0,0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,
710      0,0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,
720      0,0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,
730      0,0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,
740      0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,0,
750      0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,0,
760      0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,
770      0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,
780      0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,
790      0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,
800      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
810 }
820 dim char dia(255)={
830      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
840      0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,
850      0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,
860      0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,
870      0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,
880      0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,
890      0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,0,
900      0,0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,
910      0,0,1,1,1,1,1,1,1,1,1,1,1,0,0,0,
920      0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,0,
930      0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,
940      0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,
950      0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,
960      0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,
970      0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,
980      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
990 }
1000 dim char clb(255)={
1010     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
1020     0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,
1030     0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,
1040     0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,
1050     0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,
1060     0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,
1070     0,0,1,1,1,0,0,1,0,0,1,1,1,0,0,0,
1080     0,1,1,1,1,1,0,1,0,1,1,1,1,1,0,0,
1090     0,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,
1100     0,1,1,1,1,1,0,1,0,1,1,1,1,1,0,0,
1110     0,0,1,1,1,0,0,1,0,0,1,1,1,0,0,0,
1120     0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,
1130     0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,
1140     0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,
1150     0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,
1160     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
1170 }
1180 for i=0 to 255
1190   if spd(i)=0 then c1(i)=hi(i) else c1(i)=1
1200   if hrt(i)=0 then c2(i)=hi(i) else c2(i)=5
1210   if dia(i)=0 then c3(i)=hi(i) else c3(i)=5
1220   if clb(i)=0 then c4(i)=hi(i) else c4(i)=1
1230 next
1240 print "★";
1250 sp_def(1,c1,1):sp_def(2,c2,1)
1260 sp_def(3,c3,1):sp_def(4,c4,1)
1270 dim char ace(255)
1280 symbol(0,0,"A",1,1,1,1,0)
1290 get(0,0,15,15,ace):wipe()
1300 dim char two(255)
1310 symbol(0,0,"2",1,1,1,1,0)
1320 get(0,0,15,15,two):wipe()
1330 dim char tre(255)
1340 symbol(0,0,"3",1,1,1,1,0)
1350 get(0,0,15,15,tre):wipe()
1360 dim char fur(255)
1370 symbol(0,0,"4",1,1,1,1,0)
1380 get(0,0,15,15,fur):wipe()
1390 dim char fiv(255)
1400 symbol(0,0,"5",1,1,1,1,0)
1410 get(0,0,15,15,fiv):wipe()
1420 dim char six(255)
1430 symbol(0,0,"6",1,1,1,1,0)
1440 get(0,0,15,15,six):wipe()
1450 dim char svn(255)
1460 symbol(0,0,"7",1,1,1,1,0)
1470 get(0,0,15,15,svn):wipe()
1480 dim char ete(255)
1490 symbol(0,0,"8",1,1,1,1,0)
1500 get(0,0,15,15,ete):wipe()
1510 dim char nin(255)
1520 symbol(0,0,"9",1,1,1,1,0)
1530 get(0,0,15,15,nin):wipe()
1540 dim char ten(255)
1550 symbol(1,0,"1",1,1,1,1,0)
1560 symbol(7,0,"0",1,1,1,1,0)
1570 get(0,0,15,15,ten):wipe()
1580 dim char jak(255)
1590 symbol(0,0,"J",1,1,1,1,0)
1600 get(0,0,15,15,jak):wipe()
1610 dim char qen(255)
1620 symbol(0,0,"Q",1,1,1,1,0)
1630 get(0,0,15,15,qen):wipe()
1640 dim char kng(255)
1650 symbol(0,0,"K",1,1,1,1,0)
1660 get(0,0,15,15,kng):wipe()
1670 for j=0 to 1
1680   k=j*4+1
1690   for i=0 to 255
1700     if ace(i)=0 then c1(i)=lw(i) else c1(i)=k
1710     if two(i)=0 then c2(i)=lw(i) else c2(i)=k
1720     if tre(i)=0 then c3(i)=lw(i) else c3(i)=k
1730     if fur(i)=0 then c4(i)=lw(i) else c4(i)=k
1740     if fiv(i)=0 then c5(i)=lw(i) else c5(i)=k
1750     if six(i)=0 then c6(i)=lw(i) else c6(i)=k
1760     if svn(i)=0 then c7(i)=lw(i) else c7(i)=k
1770     if ete(i)=0 then c8(i)=lw(i) else c8(i)=k
1780     if nin(i)=0 then c9(i)=lw(i) else c9(i)=k
1790     if ten(i)=0 then ct(i)=lw(i) else ct(i)=k
1800     if jak(i)=0 then cj(i)=lw(i) else cj(i)=k
1810     if qen(i)=0 then cq(i)=lw(i) else cq(i)=k
1820     if kng(i)=0 then ck(i)=lw(i) else ck(i)=k
1830   next
1840   print "★";
1850   sp_def(j*13+5 ,c1,1):sp_def(j*13+6 ,c2,1)
1860   sp_def(j*13+7 ,c3,1):sp_def(j*13+8 ,c4,1)
1870   sp_def(j*13+9 ,c5,1):sp_def(j*13+10,c6,1)
1880   sp_def(j*13+11,c7,1):sp_def(j*13+12,c8,1)
1890   sp_def(j*13+13,c9,1):sp_def(j*13+14,ct,1)
1900   sp_def(j*13+15,cj,1):sp_def(j*13+16,cq,1)
1910   sp_def(j*13+17,ck,1)
1920 next
1930 dim char jkr(255)={
1940     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
1950     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
1960     0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,
1970     0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,
1980     0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,
1990     0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,
2000     0,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,
2010     0,1,0,0,0,0,1,0,0,1,1,1,1,0,0,0,
2020     0,1,0,0,0,0,1,0,0,0,0,0,1,0,0,0,
2030     0,1,0,0,0,0,1,0,0,0,0,0,1,0,0,0,
2040     0,1,0,0,0,0,1,0,0,1,1,1,1,0,0,0,
2050     0,1,1,1,1,1,0,0,1,0,0,0,1,0,0,0,
2060     0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,
2070     0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,0,
2080     0,0,0,0,0,0,0,0,0,1,1,0,1,0,0,0,
2090     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2100 }
2110 for i=0 to 255
2120   if jkr(i)=0 then {
2130     c1(i)=hi(i):c2(i)=lw(i)
2140   } else {
2150     c1(i)=5:c2(i)=5
2160   }
2170 next
2180 print "★";
2190 sp_def(31,c1,1):sp_def(32,c2,1)
2200 dim char bu(255)={
2210     0, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0, 0,
2220     3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0,
2230     3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0,
2240     3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0,
2250     3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0,
2260     3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0,
2270     3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0,
2280     3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0,
2290     3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0,
2300     3, 3,13,13,13,13,13, 3,13,13,13,13,13,13, 3, 0,
2310     3, 3,13, 3, 3, 3,13, 3,13, 3, 3, 3, 3,13, 3, 0,
2320     3, 3, 3,13, 3, 3, 3,13, 3,13, 3, 3, 3,13, 3, 0,
2330     3, 3, 3,13, 3, 3, 3,13, 3,13, 3, 3,13, 3, 3, 0,
2340     3, 3, 3, 3,13, 3, 3, 3,13, 3,13, 3,13, 3, 3, 0,
2350     3, 3, 3, 3,13, 3, 3, 3,13, 3,13,13, 3, 3, 3, 0,
2360     3, 3, 3, 3, 3,13, 3, 3, 3,13, 3,13, 3, 3, 3, 0
2370 }
2380 dim char bl(255)={
2390     3, 3, 3, 3, 3,13, 3, 3, 3,13, 3, 3, 3, 3, 3, 0,
2400     3, 3, 3, 3,13, 3,13, 3, 3, 3,13, 3, 3, 3, 3, 0,
2410     3, 3, 3,13,13, 3,13, 3, 3, 3,13, 3, 3, 3, 3, 0,
2420     3, 3,13, 3, 3,13, 3,13, 3, 3, 3,13, 3, 3, 3, 0,
```

```
2430     3, 3,13, 3, 3,13, 3,13, 3, 3, 3,13, 3, 3, 3, 0,
2440     3,13, 3, 3, 3, 3,13, 3,13, 3, 3, 3,13, 3, 3, 0,
2450     3,13, 3, 3, 3, 3,13, 3,13, 3, 3, 3,13, 3, 3, 0,
2460     3,13,13,13,13,13,13,13, 3,13,13,13,13, 3, 3, 0,
2470     3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0,
2480     3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0,
2490     3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0,
2500     3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0,
2510     3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0,
2520     3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0,
2530     3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0,
2540     0, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0, 0
2550 }
2560 sp_def(33,bu,1):sp_def(34,bl,1)
2570 dim char icn(255)={
2580     0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
2590     0, 0, 0, 0, 0, 1,15, 1, 0, 0, 0, 0, 0, 0, 0, 0,
2600     0, 0, 0, 0, 0, 1,15, 1, 0, 0, 0, 0, 0, 0, 0, 0,
2610     0, 0, 0, 0, 0, 1,15, 1, 0, 0, 0, 0, 0, 0, 0, 0,
2620     0, 0, 0, 0, 0, 1,15, 1, 0, 0, 0, 0, 0, 0, 0, 0,
2630     0, 0, 0, 0, 0, 1,15, 1, 1, 0, 0, 0, 0, 0, 0, 0,
2640     0, 0, 0, 0, 0, 1,15, 1,15, 1, 1, 0, 0, 0, 0, 0,
2650     0, 0, 1, 1, 0, 1,15, 1,15, 1,15, 1, 1, 0, 0, 0,
2660     0, 0, 1,15, 1, 1,15, 1,15, 1,15, 1,15, 1, 0, 0,
2670     0, 0, 1,15,15, 1,15, 1,15, 1,15, 1,15, 1, 0, 0,
2680     0, 0, 0, 1,15,15,15,15,15,15,15, 1,15, 1, 0, 0,
2690     0, 0, 0, 1,15,15,15,15,15,15,15,15,15, 1, 0, 0,
2700     0, 0, 0, 0, 1,15,15,15,15,15,15,15,15, 1, 0, 0,
2710     0, 0, 0, 0, 1,15,15,15,15,15,15,15, 1, 0, 0, 0,
2720     0, 0, 0, 0, 0, 1,15,15,15,15,15,15, 1, 0, 0, 0,
2730     0, 0, 0, 0, 0, 1,15,15,15,15,15,15, 1, 0, 0, 0
2740 }
2750 sp_def(0,icn,1):sp_color(1,1)
2760 end
```

リスト2 メインプログラム

```
  10 /*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*
  20 /*                                      /*
  30 /*      ーーババぬきだよーん!ーー        /*
  40 /*            by T.Mounai               /*
  50 /*                                      /*
  60 /*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*
  70 /*
  80 dim int card(52),ctbl(27),prc(4),dcd(4)
  90 dim str mg(54)
 100 int i,x,demo=0
 110 /*
 120 mouse(2):mouse(4)
 130 m_init():m_alloc(1,8000):m_assign(1,1)
 140 m_trk(1,"@38o6v15c32")
 150 screen 0,0,1,1:cls
 160 locate 7,0:print"ババ抜きで遊ぼう"
 170 mgset()
 180 randomize(val(right$(time$,2)))
 190 while 1
 200   spcls()
 210   sfl()
 220   init()
 230   game()
 240   pause(20)
 250 endwhile
 260 end
 270 /*
 280 func init()
 290 for x=1 to 13
 300   cdprt(x,3,5,0):pause(1)
 310   cdprt(x,0,5,0):pause(1)
 320   cdprt(x,1,5,0):pause(1)
 330   cdprt(x,2,5,0):pause(1)
 340 next
 350 cdprt(14,3,5,0):pause(10)
 360 for i=0 to 13
 370   cdprt(i+1,3,getst(card(i+39)),getnum(card(i+39)))
 380 next
 390 for i=0 to 12:ctbl(i)=card(i):next:sute(0)
 400 for i=0 to 12:card(i)=ctbl(i):next
 410 for i=0 to 12:ctbl(i)=card(i+13):next:sute(1)
 420 for i=0 to 12:card(i+13)=ctbl(i):next
 430 for i=0 to 12:ctbl(i)=card(i+26):next:sute(2)
 440 for i=0 to 12:card(i+26)=ctbl(i):next
 450 for i=0 to 13:ctbl(i)=card(i+39):next:sute(3)
 460 for i=0 to 13:card(i+39)=ctbl(i):next:pause(15)
 470 endfunc
 480 /*
 490 func cdprt(x,y,st,num)
 500 int page
 510 switch y
 520   case 0:page= 1+x*2:break
 530   case 1:page=30+x*2:break
 540   case 2:page=60+x*2:break
 550   case 3:page=90+x*2:y=y+1
 560 endswitch
 570 switch st
 580   case 1:num=num+13:break
 590   case 2:num=num+13:break
 600   case 4:st=30:num=28:break
 610   case 5:st=32:num=30:break
 620   case 6:st=34:num=32:break
 630 endswitch
 640 y=y+1
 650 sp_move(page,x*16,y*34,st+1)
 660 sp_move(page+1,x*16,y*34+16,num+4)
 670 endfunc
 680 /*
 690 func spcls()
 700 for i=0 to 127:sp_move(i,0,0,35):next
 710 sp_disp(1)
 720 endfunc
 730 /*
 740 func getst(cd)
 750 return((cd-1)/13)
 760 endfunc
 770 /*
 780 func getnum(cd)
 790 return((cd-1) mod 13+1)
 800 endfunc
 810 /*
 820 func sfl()
 830 int p1,p2,bf
 840 for i=1 to 53:card(i-1)=i:next
 850 for i=0 to 120
 860   p1=rnd()*53:p2=rnd()*53
 870   bf=card(p1)
 880   card(p1)=card(p2)
 890   card(p2)=bf
 900 next
 910 endfunc()
 920 /*
 930 func pause(cnt)
 940 int i
 950 cnt=cnt*500
 960 for i=0 to cnt:next
 970 endfunc
 980 /*
 990 func sute(y)
1000 int i,j,c1,c2
1010 for i=0 to 12
1020   for j=i+1 to 13
1030     c1=getnum(ctbl(i))
1040     c2=getnum(ctbl(j))
1050     if c1=c2 and ctbl(i)<53 and ctbl(j)<53 then {
1060       ctbl(i)=0:ctbl(j)=0
1070       cdprt(i+1,y,6,0):cdprt(j+1,y,6,0)
1080       break
1090     }
1100   next
1110 next
1120 for i=14 to 27:ctbl(i)=0:next
1130 j=0
1140 for i=0 to 13
1150   if ctbl(i)<>0 then ctbl(j+14)=ctbl(i):j=j+1
1160 next
1170 for i=0 to 13:ctbl(i)=ctbl(i+14):next
1180 for i=0 to 13
1190   if y<>3 then j=5 else j=getst(ctbl(i))
1200   if ctbl(i)=0 then j=6
1210   cdprt(i+1,y,j,getnum(ctbl(i)))
1220 next
1230 endfunc
1240 /*
1250 func game()
1260 int pr
1270 stprc()
1280 pr=4
1290 repeat
1300   play(pr)
1310   repeat
1320     pr=pr-1
1330     if pr=0 then pr=4
1340   until prc(pr)<>0
1350 until prc(1)+prc(2)+prc(3)+prc(4)=1
1360 if prc(4)=0 then koe(3):pause(50)
1370 prtmsg("")
1380 endfunc
1390 /*
1400 func play(pr)
1410 int ppr,dpr
1420 ppr=pr:dpr=ppr
1430 repeat
1440   dpr=dpr-1
1450   if dpr=0 then dpr=4
1460 until prc(dpr)<>0
1470 drow(ppr,dpr)
1480 endfunc
1490 /*
1500 func drow(ppr,dpr)
1510 int cd,cdd,ct,x,y,i
1520 if ppr<>4 or demo=1 then {
1530   koe(0)
1540   x=0:y=dpr*34+12
1550   if dpr=4 then y=y+34
1560   ct=rnd()*4+1
1570   for i=0 to ct
1580     sp_move(0,x+16,y,0):pause(3)
1590     cd=rnd()*prc(dpr):cdd=cd*16
1600     repeat
1610       if cdd>x then x=x+1
1620       if cdd<x then x=x-1
```

▶前期試験も佳境に入ったある日，自殺覚悟で，私はTETRISを買ってしまいました。帰宅した私を待っていたのは，1Mの増設RAMとDōGA CGAシステムでした。神様ってどうしてこんなに親切なんでしょう（グスッ)。　津田　智夫（21）千葉県

```
1630       sp_move(0,x+16,y,0)
1640     until cdd=x
1650   pause(3)
1660   next
1670 } else {
1680   koe(7)
1690   cd=msctl(dpr)
1700 }
1710 for i=0 to 5
1720   if dpr<>4 then {
1730     cdprt(cd+1,dpr-1,5,0)
1740   } else {
1750     cdprt(cd+1,dpr-1,getst(card(cd+39)),getnum(card(cd
+39)))
1760   }
1770   cdprt(cd+1,dpr-1,6,0)
1780 next
1790 dcd(ppr)=card(cd+13*(dpr-1))
1800   if ppr<>4 then {
1810     cdprt(14,ppr-1,5,0)
1820   } else {
1830     cdprt(14,ppr-1,getst(dcd(ppr)),getnum(dcd(ppr)))
1840   }
1850 card(cd+13*(dpr-1))=0
1860 prc(ppr)=prc(ppr)+1
1870 prc(dpr)=prc(dpr)-1
1880 if prc(dpr)=0 and dpr<>4 then koe(6):pause(5)
1890 resfl(dpr)
1900 chk(ppr)
1910 resfl(ppr)
1920 endfunc
1930 /*
1940 func chk(ppr)
1950 int ccd,cst,bcd,bst,koef,i
1960 koef=prc(ppr)
1970 ccd=getnum(dcd(ppr))
1980 cst=getst(dcd(ppr))
1990 if cst<4 then {
2000   for i=0 to prc(ppr)
2010     bcd=getnum(card(i+13*(ppr-1)))
2020     bst=getst(card(i+13*(ppr-1)))
2030     if ccd=bcd and bst<4 then {
2040       pause(10)
2050       card(i+13*(ppr-1))=0
2060       dcd(ppr)=0
2070       cdprt(i+1,ppr-1,6,0)
2080       pause(2)
2090       cdprt(14,ppr-1,6,0)
2100       if ppr<>4 then koe(2):pause(5)
2110       if prc(1)+prc(2)+prc(3)+prc(4)>1 then koe(5):pau
se(5)
2120       prc(ppr)=prc(ppr)-2
2130       if prc(ppr)=0 and ppr<>4 then koe(6):pause(5)
2140       break
2150     }
2160   next
2170 }
2180 if koef=prc(ppr) then {
2190   if ppr<>4 then koe(1):pause(5)
2200   koe(4):pause(5)
2210 }
2220 endfunc
2230 /*
2240 func resfl(plr)
2250 int i,j,c1,c2,cb,dlr
2260 j=0:dlr=plr-1
2270 for i=0 to 12
2280   cb=card(i+13*dlr)
2290   card(j+13*dlr)=card(i+13*dlr)
2300   if cb<>0 then j=j+1
2310 next
2320 if dcd(plr)<>0 then {
2330   card((prc(plr)-1)+13*dlr)=dcd(plr)
2340   dcd(plr)=0
2350 }
2360 cdprt(14,dlr,6,0)
2370 for i=0 to 12
2380   if card(i+13*dlr)=0 then {
2390     cdprt(i+1,dlr,6,0)
2400   } else {
2410     if plr<>4 then {
2420       cdprt(i+1,dlr,5,0)
2430     } else {
2440       cdprt(i+1,dlr,getst(card(i+13*dlr)),getnum(card(i
+13*dlr)))
2450     }
2460   }
2470 next
2480 if plr<>4 then {
2490   for i=0 to 9
2500     c1=rnd()*prc(plr)+13*dlr
2510     c2=rnd()*prc(plr)+13*dlr
2520     cb=card(c1)
2530     card(c1)=card(c2)
2540     card(c2)=cb
2550   next
2560 }
2570 endfunc
2580 /*
2590 func stprc()
2600 int i,j
2610 for i=1 to 4:prc(i)=0:next
2620 for i=0 to 12
2630   if card(i)<>0 then prc(1)=prc(1)+1
2640 next
2650 for i=13 to 25
2660   if card(i)<>0 then prc(2)=prc(2)+1
2670 next
2680 for i=26 to 38
2690   if card(i)<>0 then prc(3)=prc(3)+1
2700 next
2710 for i=39 to 51
2720   if card(i)<>0 then prc(4)=prc(4)+1
2730 next
2740 endfunc
2750 /*
2760 func msctl(dpr)
2770 int x,y,tm
2780 tm=0
2790 msarea(13,0,prc(dpr)*16+2,255)
2800 repeat
2810   tm=(tm+1) mod 500
2820   if tm=250 then koe(rnd()*3+8)
2830   mspos(x,y):sp_move(0,x,dpr*34+12,0)
2840 until msbtn(0,0,10)=-1
2850 return((x-9)/16)
2860 endfunc
2870 /*
2880 func prtmsg(msg;str)
2890 color 1
2900 locate 1,14:print space$(30)
2910 locate 1,14:print msg
2920 color 3
2930 endfunc
2940 /*
2950 func koe(pt)
2960 m_play()
2970 prtmsg(mg(int(rnd()*5+pt*5)))
2980 endfunc
2990 /*
3000 func mgset()
3010 mg(0)="よーし、当てるぞっ！"
3020 mg(1)="うーん、どれを引こうかなー"
3030 mg(2)="あれはどこかな？"
3040 mg(3)="どれにしようかな"
3050 mg(4)="これだと思うんだけどねー"
3060 /*
3070 mg(5)="ちぇっ、すてられねーなぁ"
3080 mg(6)="げげげっ、はずしたっ！！"
3090 mg(7)="いらねーのがきたなー"
3100 mg(8)="げろげろ…"
3110 mg(9)="こんなのいらねーよ"
3120 /*
3130 mg(10)="ほっほっほっ…あったもんね"
3140 mg(11)="あてたあてた、やったね！"
3150 mg(12)="いやー、あたった"
3160 mg(13)="ベィビー、あてたぜ！"
3170 mg(14)="あったあった"
3180 /*
3190 mg(15)="ま、まけた…"
3200 mg(16)="う、うう…くやしい！"
3210 mg(17)="ガーン！！　なんてこった…"
3220 mg(18)="げげっ、ばかやろー！！"
3230 mg(19)="まけた、くそー次にいくぞ！"
3240 /*
3250 mg(20)="あいつ、ババ引いたのかな？"
3260 mg(21)="ははは…やーいはずしてんの"
3270 mg(22)="ねえねえ、何を引いたの？"
3280 mg(23)="今おまえ、ばば引いたろ？"
3290 mg(24)="引きが悪いなー"
3300 /*
3310 mg(25)="捨てちゃった、なんてやつだ"
3320 mg(26)="くそー俺も当てるぞ！"
3330 mg(27)="まずい…捨てられた"
3340 mg(28)="ちくしょー、なんてこった"
3350 mg(29)="捨てた？それは許されないよ"
3360 /*
3370 mg(30)="あがり、やったね！"
3380 mg(31)="勝った勝った、悪いね"
3390 mg(32)="あがりー！へっへっ、お先に"
3400 mg(33)="はっはっはっ、これであがり"
3410 mg(34)="あがったあがった、お先に！"
3420 /*
3430 mg(35)="おまえさんの番だよ"
3440 mg(36)="何してんだ？おまえの番だぞ"
3450 mg(37)="おーい、おまえの番だぞ"
3460 mg(38)="ねえ、君の番だよ"
3470 mg(39)="待ってんだから早く引いてよ！"
3480 /*
3490 mg(40)="おい、カードをのぞくなよ！"
3500 mg(41)="ねえ、ジュースないの？"
3510 mg(42)="あ、そこのせんべい取って"
3520 mg(43)="うーん、おなかがすいたなー"
3530 mg(44)="ねえ、カセットでも聞こうよ"
3540 /*
3550 mg(45)="あー、ミカンが食べたいなー"
3560 mg(46)="たまには部屋の掃除でもしろよな"
3570 mg(47)="そう言えば、日曜日は雨だってさ"
3580 mg(48)="このカップラーメン食べていい？"
3590 mg(49)="あれ、もう酒がないぞ"
3600 /*
3610 mg(50)="ねえ、こんど浅香唯のＣＤ貸して"
3620 mg(51)="オレ、きのうも徹夜したんだぜ"
3630 mg(52)="おい、そのサキイカ俺のだぞ！"
3640 mg(53)="おい、明日のテスト大丈夫か？"
3650 mg(54)="ねえ、今月のＯｈ！Ｘ買った？"
3660 endfunc
```

▶車を買いました。カムリかビスタ，値段のカローラと悩んでましたがファミリア1800GT-Xの新車発表会を見て4WD，DOHC I.C.TURBO，180psのスペックにほれてしまいました。PS.トヨタの営業マンは凄い。1日3回家にくるわ，試乗車を家に持ってくるわ，値段もどんどん落すわ，マツダさんもっとがんばってください。　下沢　弘（29）北海道

★(で)のショートプロぱーてぃ その3

BLACK JACK とCROSS SHOT

Komura Satoshi 古村　聡

カードが舞い，ビームが飛ぶ！　今月は入力しやすいお手軽サイズで2度おいしい，X1のショートプログラムの2本立てというわけです。

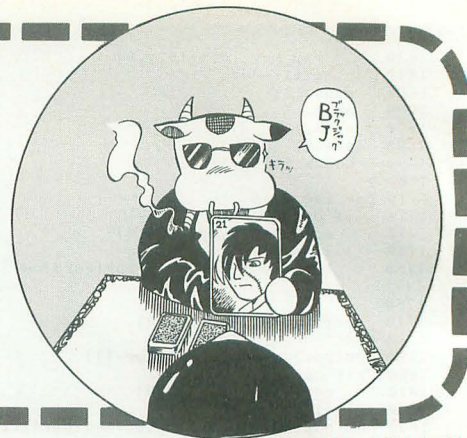



illustration : T.Takahashi

今月も2本立てだぁっ！

毎回毎回，BASICリストのてんこ盛りになってしまうこのコーナー，今月もやってしまいました，またしてもの2本立て，つーても，今月はX1が2本だから，ま，（その1）がMZ 2本立てだったことを考えるとちょうどいいのかもしれないですけどね。あとはX68000とMZ-700/1500あたりが2本あれば完璧ですね（おっと，S-OSもあったか）。

さあ，2本立てでショートを送ってくれれば，

掲載率が高い！

かもしれない。さぁ，出すならいまだ，なんてね（こうやって投稿のお誘いをかけてしまう最近の私）。

冗談はともかく，今月掲載の2本を投稿してくれた七浦くんのパワーは凄かった。今月掲載の2本を含めて1カ月でプログラム4本投稿！

だいたい今月のプログラムだって本当は掲載作品を2本にしようなんてぜーんぜん思ってなかったんですよね。ああ，それなのにそれなのに，2本が最終候補に残り，どっちを切ろうかという段になって編集担当さんに2本をプレイしてもらったら「面白いから2本とも載っけなさい」という鶴の一声で突然2本立てになってしまったという経緯があったりするのだ（このあと2本とも同じ人の作品だと知ってびびった私だった）。

うーむ，このページの掲載プログラムをむりやり2本にしたあげく，2本とも自分のプログラムをのせてしまうとは，恐るべし，七浦パワー。

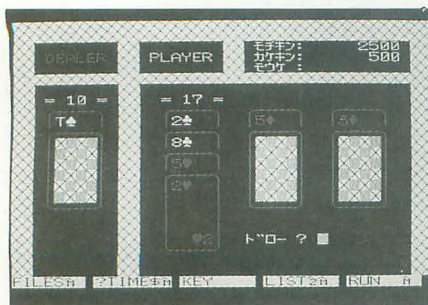
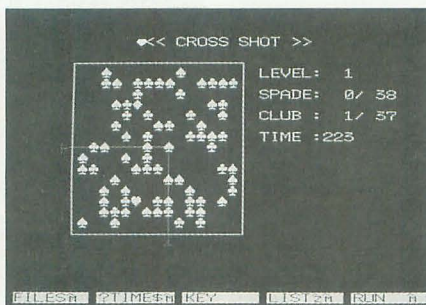
スペード，ダイヤ，ヘイヘイヘヘイっと！

さて，今月の作品の紹介に入るわけだけど，まず今月の1本目。

「BLACK JACK」

兵庫県　七浦　啓有

例のカードゲーム，俗に21ともいうあれですね。早い話が3組のカードのなかからひとつ選んで賭け金をかけて「自分のカードの合計≦21」になるように（絵札(T)は10点，エースは1，11の好きなほうの点数で数える）して親に勝てば金がもうかるというルールなわけです。

このゲームの場合ドロー（もう1枚カードを加えること）するかしないかはY/Nのキーで選ぶようになってます。なお，プレイヤー，親どちらも21点を超えるとその時点で負け。ま，基本的にはトランプゲームのルールと同じですんで，この説明でルールがまだわからなかったら知ってる人に聞いてください（もっともプログラムを読んでルールを理解してしまうのがその筋のあり方なのかもしれないけど）。

で，勝負に勝った場合の賭け金の精算について（負けたらとーぜん，賭け金は親に取られちゃうかんね）なのですが，

4枚以下のカードで勝つと，
　　賭け金＋賭け金と同額のチップ
5枚で賭け金＋賭け金の2倍のチップ
6枚で賭け金＋賭け金の4倍のチップ
7枚で賭け金＋賭け金の8倍のチップ
　　⋮

が返ってきます（もうけの欄に出る）。

そうそう，手札が3枚で，6，7，8だとチップが2倍に，7，7，7だとチップは3倍になります。

縦と横から狙い撃ち！

で，2本目の作品の紹介（ああ，2本立ては忙しい）。

「CROSS SHOT」

兵庫県　七浦　啓有

2，4，6，8のキーで青と赤の2台のレーザーを交差させるとそこにある障害物が爆発します。砲台を操って「スペード」に当ててください！

た・だ・し，時間制限があり一定時間内に命中率70%を超えないと次の面に進めず，さらに40%未満だと1レベル下の面に戻されてしまいます。そのうえ「スペード」以外のものを撃つと，

「クラブ」減点
「ダイヤ」50TIMEの間レーザーが使用不能になる

というペナルティがつきます。しかし，

「ハート」30TIMEが増える

というぐあいに捨てる神あれば，拾う神もあるゲームであったりもします。

どっちにしてもTIMEが減るのがあっという間のとーってもスリリングなゲームですので，「オムライスの食べすぎで胃のむかむかする人および運動神経がナマケモノ並みの人のご使用はご遠慮ください（私のこ

ブラックジャック

クロスショット

とだ)」なゲームです(あー疲れた)。

プログラムについて

さーて,ではこのプログラム,これからショートを作ろうという人には参考になると思うのでよーく読んでくれるとうれしかったりします。

ちなみに,

BLACK JACKの変数

変数	説明
CA (51)	……カードを使用したかどうかのフラグ
CC (13)	……カードに対する点数
TK (1)	……手札の数
TP (1)	……手札の合計点数
A (3)	……カードの番号記憶用
TN	……選んだ台札
X,Y	……カードの座標
AK	……手札のエースの数(点数が11のもの)
G#	……チップの所持数
K#	……賭け金

CROSS SHOTの変数

変数	説明
X,Y	……レーザーの座標
T	……タイム
LV	……レベル
SP	……初期スペード数
P1	……撃ったスペード数
CL	……初期クラブ数
P2	……撃ったクラブ数
ME$	……メッセージ
F	……フラグ

だそうですので参考にしてください。

んで,クロスショットは投稿されてきた時点ではリストがちょっとごちゃごちゃしてたんで私がリストを直してしまいました。あのね,いくらショートだからって無理にリストを:でつなげて行を縮めなくてもいいんですからね! むしろ多少長くなっても掲載版のリストみたいに1行1行が短いほうが打ち込みやすいと思います。ですから投稿してくださるときには自然体で書いたプログラムでOK,ばーっちりです(そうそう,あとコントロールコードで砲台の移動をしていた部分もLOCATE文で展開してしまいました)。

さて,今月も2本立てでお送りしたわけだけど,できればそのうち3本立てをやってみたいですねぇ。そのためにも投稿作品待ってます。今月のリストを読んで力をつけてください。あ,このゲームの移植版なんてのもいいかもしれませんね(特にX68000やMZ-700ね)。

そーゆーわけで,また来月。

リスト1 ブラックジャック

```
10 ' XXXX     -- BLACK JACK --     XXXX
20 ' X\/X                          X\/X
30 ' \XX/       (C)1989/8/6        \XX/
40 ' /XX\                          /XX\
50 ' X/\X     Programed By H.N     X/\X
60 ' XXXX                          XXXX
70 '
80 '■── INITIALIZE ──■
90 INIT:WIDTH 40:CLICK OFF:KBUF OFF:DEFINT A-Z
100 PRW 255:RANDOMIZE VAL(RIGHT$(TIME$,2))
110 DIM CA(51),CC(13),X(3),TK(1),TP(1),A(3):G#=1000
120 FOR I=0 TO  3:READ  X(I):NEXT
130 FOR I=1 TO 13:READ CC(I):NEXT
140 '■── MAKE SCREEN ──■
150 CLS 4:CREV 1:COLOR 4:LINE(0,0)-(40,25),"×",BF:COLOR 7:CREV
160 RESTORE 1150:FOR I=0 TO 4:READ X,Y,X1,Y1
170 LINE(X,Y)-(X1,Y1)," ",BF:NEXT
180 FOR I=0 TO 1:TK(I)=1:TP(I)=0:NEXT
190 FOR I=0 TO 51:CA(I)=0:NEXT
200 COLOR 2:LOCATE  3,3:PRINT"DEALER"
210 COLOR 5:LOCATE 13,3:PRINT"PLAYER":COLOR 7
220 LOCATE 23,2:PRINTUSING"ﾓﾁｷﾝ:#########";G#
230 LOCATE 23,3:PRINT"ｶｹｷﾝ:":LOCATE 23,4:PRINT"ﾓｳｹ :"
240 PRW:FOR J=0 TO 3
250 R=RND*51:IF CA(R)=1 THEN 250
260 CA(R)=1:X=X(J):Y=8:GOSUB 970:A(J)=C:NEXT:AK=0
270 FOR J=0 TO 3:X=X(J):Y=10:GOSUB 1070:NEXT
280 '■── INPUT DATA ──■
290 FOR I=1 TO 3:LOCATE 7+I*8,7:PRINTUSING"NO.#";I/10:NEXT
300 LOCATE 18,20:PRINT"ﾅﾝﾊﾞﾝﾆ ｶｹﾏｽｶ ? ";:A$=INKEY$(1):PRINT A$:T
N=VAL(A$)
310 IF TN<1 OR TN>3 THEN LOCATE 18,20:PRINT SPC(17):GOTO 300
320 FOR I=7 TO 20 STEP 13:LOCATE 15,I:PRINT SPC(20):NEXT
330 LOCATE X(TN),7:PRINTUSING"= ## =";CC(A(TN))
340 LOCATE     3,7:PRINTUSING"= ## =";CC(A(0))
350 A$="":LOCATE 18,20:PRINT"ｲｸﾗ ｶｹﾏｽｶ ?"
360 I$=INKEY$:IF I$="" THEN 360
370 IF I$=CHR$(13) THEN 430
380 IF I$=CHR$(8) AND A$<>"" THEN A$=LEFT$(A$,LEN(A$)-1) ELSE400
390 LOCATE 36-LEN(A$),3:PRINT" ":GOTO 420
400 IF I$<"0" OR I$>"9" OR LEN(A$)>8 THEN 360
410 A$=A$+I$:IF LEFT$(A$,1)="0" THEN 350
420 LOCATE 37-LEN(A$),3:PRINT A$:GOTO 360
430 K#=VAL(A$):IF K#<=G# AND K#<>0 THEN 450
440 A$="":LOCATE 28,3:PRINT SPC(9):GOTO 350
450 LOCATE 18,20:PRINT SPC(11):LOCATE 28,2:PRINT SPC(9):A$=STR$(
G#-K#)
460 LOCATE 38-LEN(A$),2:PRINT RIGHT$(A$,LEN(A$)-1)
470 '■── PLAYER GAME ──■
480 IF A(TN)=1 THEN AK=1
490 TP(1)=CC(A(TN)):A(1)=A(TN)
500 R=RND*51:IF CA(R)=1 THEN 500
510 CA(R)=1:TK(1)=TK(1)+1:X=X(TN):Y=6+TK(1)*2:CONSOLE 8,16,X,7
520 IF Y>16 THEN Y=16:LOCATE X,23:PRINT CHR$(13)
530 GOSUB 970:CONSOLE:TP(1)=TP(1)+CC(C)
540 IF TK(1)<4 THEN A(TK(1))=C
550 IF TP(1)>21 AND AK>0 THEN TP(1)=TP(1)-10:AK=AK-1:GOTO 550
560 LOCATE X(TN)+2,7:PRINTUSING"##";TP(1)
570 IF TP(1)>21 THEN PAUSE 5:GOTO 850
580 LOCATE 13-(TN=1)*9,20:PRINT"ﾄﾞﾛｰ ? ";
590 A$=INKEY$(1):A=INSTR("YyNn",A$):IF A=0 THEN 590
600 LOCATE 13-(TN=1)*9,20:PRINT SPC(8)
610 IF A<3 THEN 500
620 PAUSE 10:AK=0
630 '■── DEALER GAME ──■
640 TP(0)=CC(A(0)):IF A(0)=1 THEN AK=1
650 R=RND*51:IF CA(R)=1 THEN 650
660 CA(R)=1:TK(0)=TK(0)+1:X=3:Y=6+TK(0)*2:CONSOLE 8,16,3,7
670 IF Y>16 THEN Y=16:LOCATE 3,23:PRINT CHR$(13)
680 GOSUB 970:CONSOLE:TP(0)=TP(0)+CC(C)
690 IF TP(0)>21 AND AK>0 THEN TP(0)=TP(0)-10:AK=AK-1:GOTO 690
700 LOCATE 5,7:PRINTUSING"##";TP(0)
710 IF TP(0)>21 THEN PAUSE 5:GOTO 750
720 PAUSE 5:IF TP(0)<17 THEN 650
730 '■── GAME JUDGE ──■
740 IF TP(0)>=TP(1) THEN 850
750 IF TK(1)=3 AND TP(1)=21 THEN 780
760 IF TK(1)>4 THEN K#=K#*2^(TK(1)-4)
770 GOTO 830
780 A=0:FOR I=1 TO 2:IF A(I)<=A(I+1) THEN 800
790 A=1:SWAP A(I),A(I+1)
800 NEXT:ON A GOTO 780
810 FOR I=1 TO 3:IF A(I)=7   THEN NEXT:K#=K#*3
820 FOR I=1 TO 3:IF A(I)=5+I THEN NEXT:K#=K#*2
830 LOCATE 13-(TN=1)*9,20:PRINT"YOU WIN !":G#=G#+K#
840 SOUND 7,&H3F:SOUND 8,0:PLAY"C1DEDEFEFG+C4+D+C":GOTO 870
850 COLOR 2:LOCATE 13-(TN=1)*9,20:PRINT"YOU LOSE":G#=G#-K#:K#=0
860 COLOR 7:SOUND 7,&H3F:SOUND 8,0:PLAY"+C1GFDEC"
870 IF LEN(STR$(K#))>10 THEN K#=999999999#
880 IF LEN(STR$(G#))>10 THEN G#=999999999#
890 LOCATE 38-LEN(STR$(K#)),4:PRINT RIGHT$(STR$(K#),LEN(STR$(K#)
)-1)
900 PAUSE 10:IF G#>0 THEN 160
910 '■── GAME OVER ──■
920 CREV 1:COLOR 3:LINE(15,10)-(33,16),"▚",B:CREV:LINE(16,11)-(3
2,15)," ",BF
930 COLOR 6:LOCATE 20,12:PRINT"GAME OVER"
940 FOR I=1 TO 7:COLOR I:LOCATE 18,14:PRINT"HIT SPACE KEY"
950 IF INKEY$=" " THEN CLS:RUN ELSE NEXT:GOTO 940
960 '■── CARD PRINT ──■
970 C=R MOD 13+1:B=R¥13:IF C=1 THEN AK=AK+1
980 COLOR 1:LOCATE X,Y:PRINT"╭───╮"
990 FOR I=1 TO 6:LOCATE X,Y+I:PRINT"│     │":NEXT
```

```
1000 LOCATE X,Y+7:PRINT"└───┘";:A$=MID$("♠♥♦♣",B+1,1)
1010 IF C=1 OR C>9 THEN B$=MID$("ATJQK",1-(C>9)*(C-9),1):GOTO 10
30
1020 B$=RIGHT$(STR$(C),1)
1030 COLOR 7+(B=1 OR B=2)*5
1040 LOCATE X+1,Y+1:PRINT B$;A$
1050 LOCATE X+3,Y+6:PRINT A$;B$
1060 GOSUB 1110:RETURN
1070 COLOR 1:LOCATE X,Y:PRINT"┌───┐"
1080 FOR I=1 TO 6:IF I MOD 2=0 THEN A$="▒×▒×" ELSE A$="×▒×▒"
1090 LOCATE X,Y+I:PRINT"|";:CREV1:COLOR 6:PRINT A$;:CREV:COLOR 1
1100 PRINT"|":NEXT:LOCATE X,Y+7:PRINT"└───┘":GOSUB 1110:RETURN
1110 SOUND 6,13:SOUND 7,55:SOUND 8,16:SOUND 11,0:SOUND 12,7:SOUN
D 13,57
1120 COLOR 7:RETURN
1130 DATA 3,14,22,30
1140 DATA 11,2,3,4,5,6,7,8,9,10,10,10,10
1150 DATA 2,2,9,4,12,2,19,4,22,2,37,4,2,6,9,23,12,6,37,23
```

ぎっちり詰まった元のリスト

リスト2 クロスショット

```
10 INIT:CLS4:WIDTH 40
20 DEFINT A-Z:CLICKOFF:PRW 255:T=400:LV=1:X=15:Y=10
30 ME$(0)="BACK TO DOWN LEVEL":ME$(1)="TRY AGAIN !!":ME$(2)="LET
'S GO NEXT LEVEL"
40 COLOR5
50 A$=STRING$(15,144)
60 LOCATE5,2:PRINT"┌";A$;"┐"
70 FOR I=3 TO 17
80   LOCATE 5,I
90     PRINT"|";SPC(15);"|"
100 NEXT
110 LOCATE5,18
120 PRINT"└";A$;"┘"
130 FOR I=1 TO 2
140   FOR J=1 TO 35+RND*10
150     COLOR7:V=6+RND*14:W=3+RND*14
160     IF SCRN$(V,W,1)<>" " THEN 150
170     LOCATE V,W:A$=MID$("♠♣",I,1)
180     PRINTA$
190     SP=SP-(A$="♠"):CL=CL-(A$="♣")
200   NEXT
210 NEXT
220 P1=0:P2=0
230 LOCATE 12,0:PRINT"<< CROSS SHOT >>"
240 FOR I=1 TO 2
250   FOR J=1 TO 3
260     V=6+RND*14:W=3+RND*14
270     IF SCRN$(V,W,1)=" " THEN LOCATE V,W:PRINT MID$("♥♦",I,1)
280   NEXT
290 NEXT
300 LOCATE 23,3:PRINTUSING"LEVEL: ##";LV
310 LOCATE 23,5:PRINT"SPADE:  0/";SP
320 LOCATE 23,7:PRINT"CLUB :  0/";CL
330 LOCATE 23,9:PRINTUSING"TIME :###";T:PRW
340 REPEAT
350   S=STICK(0)+STICK(1)
360   X=X+((S=4 AND X>6)-(S=6 AND X<20))
370   Y=Y+((S=8 AND Y>3)-(S=2 AND Y<17))
380   A$=SCRN$(X,Y,1):IF F-50=T THEN F=0
390   COLOR1:LOCATE X-1,19:PRINT" ┴ ":COLOR2
400   LOCATE 4 ,Y-1:PRINT" ";
410   LOCATE 4 ,Y  :PRINT"├";
420   LOCATE 4 ,Y+1:PRINT" ";:COLOR7
430   IF (STRIG(0)+STRIG(1))*(F=0)=1    ELSE 600
440 '     {
450      V=51+(X-6)*8:W=35+(Y-4)*8
460      LINE(V,150)-(V,19-(A$<>" ")*(Y-2)*8),PSET,1
470      LINE(41,W)-(171+(A$<>" ")*(21-X)*8,W),XOR,2
480      IF A$<>" "                          ELSE CLS0:GOTO 590
490 '        {
500         P1=P1-(A$="♠")
510         P2=P2-(A$="♣")
520         T=T-(A$="♥")*31
530         F=-(A$="♦")*T
540         LOCATE X,Y
550         COLOR INSTR("♣♠♥♦",A$)
560         PRINT"*"
570         CLS0:BEEP
580         LOCATE X,Y:PRINT" ":COLOR7
590 '        }
600 '     }
610   T=T-1
620   LOCATE 30,5:PRINTUSING"##";p1
630   LOCATE 30,7:PRINTUSING"##";P2
640   LOCATE 29,9:PRINTUSING"###";T
650 UNTIL t=0 OR p1=sp
660 CFLASH 1:LOCATE 25,12
670 LOCATE 25,12
680 IF P1<>SP THEN A$="TIME UP"
         ELSE IF P2<>0 THEN A$="VERY GOOD"
                        ELSE A$="PERFECT"
690 ' END IF
700 PRINT A$;" !!":CFLASH:PAUSE 10
710 A=(P1-P2)/SP*100:IF A<0 THEN A=0
720 LOCATE 23,14:PRINT"ﾀﾀﾞｲﾏﾉ ﾒｲﾁｭｳﾘﾂﾊ ":PAUSE 10
730 LOCATE 24,16:PRINT A;"% ﾃﾞｼﾀ"
740 A=-(A>39 OR LV=1)-(A>69)
750 LOCATE 20-LEN(ME$(A))/2,22:PRINT ME$(A):PAUSE 50
760 LV=LV+A-1:T=400-(LV-1)*30:X=14:Y=9:F=0:SP=0:CL=0
770 CLS:PRW 255:COLOR5:GOTO 50
```

THE SENTINEL

●8ビットの逆襲

「最近誌面がどーもX68000に喰われてしまっているような気がする」,「Oh！XはX68000だけの雑誌ではない」というお便りを目にすることが多い今日このごろです。そりゃ8ビットの64Kバイトのメモリに比べればX68000の1Mバイトはおいしいし，グラフィックはジョーダンじゃないくらい凄いし，スプライトは……。そう，確かにマシンの機能だけを見れば，天と地ほどの差があります。でも，プログラマの心意気ってそんなものではないと思うのです。

S-OSはフロンティア精神旺盛な，プログラマの心意気によって支えられてきた企画です。最近市販されるゲームなどを見て思うのは，この人たちは作っていて面白いのだろうかという疑問です。グラフィックは凄い。サウンドも並じゃない。でもゲームの基幹をなすアイデアは，どこかで見たことがあるようなものばかり。なんだか重箱の隅をつつき合っていて，皮だけが違う饅頭が山のようにあふれているのではないか。そんな気がします。先月発表したTTI用のゲームができました。緑やピンクのアンマンでなく，カレーマンのようなこのゲーム，楽しんでみてください。

第84部
パズルゲーム PUSH BON!

●パズルゲームPUSH BON！

思考型パズルゲームにはいろいろな形のものがあります。障害物を押しのけて（あるいは回避して）目的の物を手に入れるというのもそのひとつです。このPUSH BON！は，アーケードゲームのペンゴと，パソコンゲームの倉庫番を足して2で割ったようなゲームです。

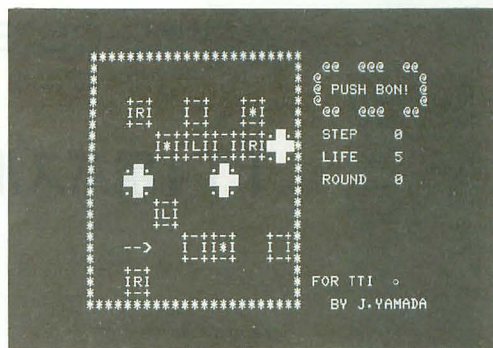
自分自身を表す矢印を4方向に移動させ，目的のブロックを押しのけて，マークの付いているブロックを1列に並べればいいという単純なルールながら，ひとひねり加わっていることにより，ひと筋縄ではいかない難しさがあります。それは，

1）押しのけたブロックは，別のブロックか壁に当たるまで押した方向に移動し続ける。

2）押しのけたブロックにL(Left)とかR(Right)とか書いてある場合は，そのブロックに当たったブロックが，書いてある文字の方向に飛ばされる。

という2つのルールによります。

ま，いいや，とりあえずやってみよう。と思って安易に始めたところ，たちまち身動きがとれなくなってしまいました。

秋の夜長にぴったりのゲームです。ぜひプレイしてみてください。

●S-OSの系譜(4)

読者投稿の第2弾となったのは，当時の情報処理試験でアセンブラの対象となっていたCAP-Xを対象としたCAP-X85です。

このプログラムは仮想上のターゲットマシンであるCOMP-Xのシミュレーションを行う部分と，COMP-Xのアセンブリ言語をCOMP-Xのマシン語に変換するアセンブラであるCAP-X，そして，プログラムを書くためのエディタがセットになったものです。このプログラムのおかげで，情報処理試験第2種のアセンブラは完璧だったといううれしいお便りも届いたものです。

また，12月号では論理型プログラミング言語として有名なPrologが発表されました。Prologはこれまでも，BASIC版，BASIC＋マシン語版(MZ-1500用)が発表されていたので，S-OS対応版がいつ出るか，期待されていたところです。プログラムの実行手順を記述するのではなく，目的を達成するための条件だけを書いていけば，コンピュータが勝手に推論してくれるPrologには独特の味があり，ファンも多いようです。Prolog-85に付属するエディタは，ZEDAなどで採用されていた1文字のコマンドから，BASICのようにわかりやすいコマンドへと変更されており，しかも，その省略型までサポートされるという本格的なものでした。

S-OSはソフトウェア資産のない，まったくゼロの状態からスタートしたわけですが，ここに至ってLisp，Prolog，という人工知能指向の2つの高級言語，アセンブラ，デバッガ，ソースコードジェネレータというマシン語の3つの神器などなど，共通システムとして本格的なアプリケーションが揃ってきました。

ソフトのないうちは信用できないと，S-OSの動向をしばらく黙って見ていた読者の方のバックナンバーの注文が相次ぎ，Oh！MZ史に残る早さでバックナンバーが在庫切れを起こしたのもこのころです。定価の2倍，3倍のプレミアムがついての取り引きまで現れ，S-OSの再掲載あるいはバージョンアップ版の早期発表が望まれました。

TTI用パズルゲーム PUSH BON!

今回は，前回作成したTTI用のパズルゲームの紹介です。TTI特有の拡張命令を多用していますのでTTCとの比較などの学習にも役立つでしょう。また，ゲーム自体もなかなかやりごたえのあるものになりましたので，ぜひ入力してみてくださいね。

Yamada Junji
山田 純二

TTI用パズルです

TTCのインタプリンタ版であるTTI用のパズルゲーム「PUSH BON」ができました。ゲーム内容は，矢印を使ってフィールド内のブロックをはじき飛ばし，"＊"マークのついたブロックを3つ，縦か横に並べると面クリアというものです。よくわからない人はちょっとペンゴを複雑にしたような感じだと思っておいてください。

とこれだけの説明だと簡単そうに思われるでしょうが実際には，なかなかハードでやりごたえのあるゲームになっています。はたして，この言葉が真実かどうか確かめたかったら，入力して遊んでみてね。

入力方法とキー操作

メインプログラムは素直にTTIのエディタで入力すればよいのですが，E-MATEなど別のエディタを使用する場合は，必ず実行前にXコマンドでテキスト格納先頭アドレスの設定をやり直すのを忘れないでください。面データはオブジェクトのかたちになっていますのでMACINTO-Cなどのツールを使って入力してください。実行方法は面データをロード後TTIに移りテキストをロードしてから，

G［リターン］

で実行できます。

このプログラムはTTIの拡張命令を多用しているので，このままではTTCでコンパイルできません。速度的にはインタプリタでもほとんど問題ないと思われますが，気にいらない人は各自で展開していってください。

キー操作はKを中心としたI, M, J, Lキーで，矢印の上下左右の移動，スペースキーで矢印の前にあるブロックをPUSHします。そしてGキーでギブアップ！　行き詰まってどうにもならなかったときに，面の初めからやり直します。ギブアップするたびにLIFEがひとつずつ減っていきLIFEが0になるとゲームオーバーとなります。

あとSTEPというのがあって，これはブロックを1回PUSHすると，ひとつずつカウントされていき255回で1回ギブアップとなります。

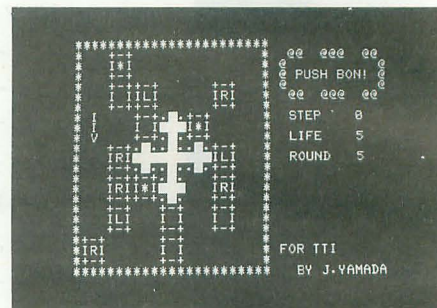
キー入力は（ラベル名で）13行目からですので，自分の機種で使いやすいように書き換えておくとよいでしょう。プログラムを改造するときには，変数の受け渡しや仮想VRAMのアドレスなどに気をつけるようにしてください。

ルールの説明

それではゲームのルールの解説をしていきましょう。まずブロックは全部で5種類あります。十字型をしたブロックは障害物で動かすことはできません。

"　"，"＊"マークのブロックはふつうのブロック。で次にR, Lマークの書かれたブロックが曲者なんです。

まず，R, Lのブロックをほかのブロックに当てた場合は(図1)，当てられたほうのブロックがR, Lブロックの押されてきた方向に対して右または左方向にはじき飛ばされます。

今度は逆に"　"，"＊"マークのブロックがR, Lマークのブロックに当たった場合は(図2)，押されてきたほうのブロックが自分の進んできた方向に対して右または左方向にはじき飛ばされます。

この規則はブロックが10回以上跳ね返るか，行き場がなくなるまで働き続けます。また，静止して隣りあったブロックもこの規則で動かすことができます。

ここのところが少し複雑でこのゲームの面白いところなんです。一応，10面に練習面を作っておきましたので，思う存分自分で納得のいくまでブロックをつつきまくってください。

以上で，ゲームの説明も終わりです。あとは読者の皆さんがそれぞれ遊んでくれて，愛読者ハガキにひと言でもいいからこのゲームの感想でも書いてくれると非常にうれしいですね。

パズルゲームを作ろう

パズルゲームを作る楽しさはなんといっても，他人にプレイしてもらって自分の作り上げた不条理な世界でもがくさまを見ることでしょう。ちょっと陰険かもしれませんが，他人が困っている姿を見るのは楽しいものです。

そして，8月号で華門氏もいってましたが，パズルゲームとは発想の問題なので，きちんと規則をまとめていけば初心者にも必ず完成させることができるはずです。しかし，気をつけなければならないのは，完成したルールが自分にしか理解できない，もしくは説明に時間がかかりすぎてしまうようなことにならないようにすることです。

表1　変数表

X, Y	カーソルの座標
S	ステップ数
D	カーソルの方向
R	ラウンド
G	残りライフ
9F00H～	仮想VRAM
A000H～	面データ

図1

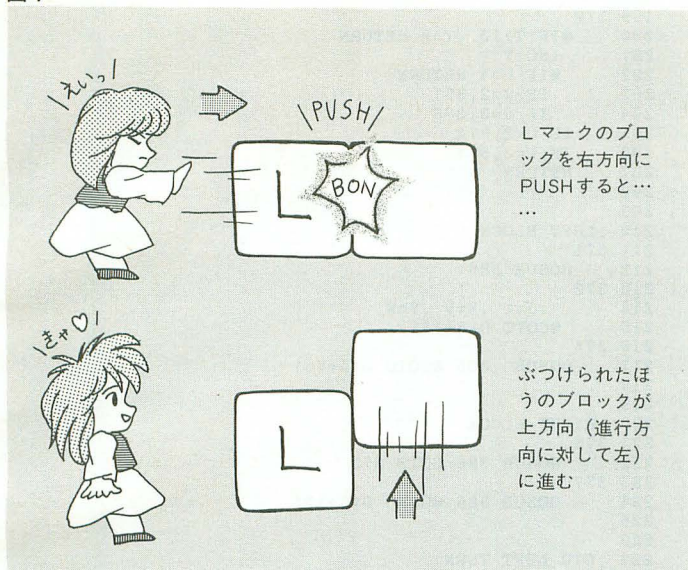
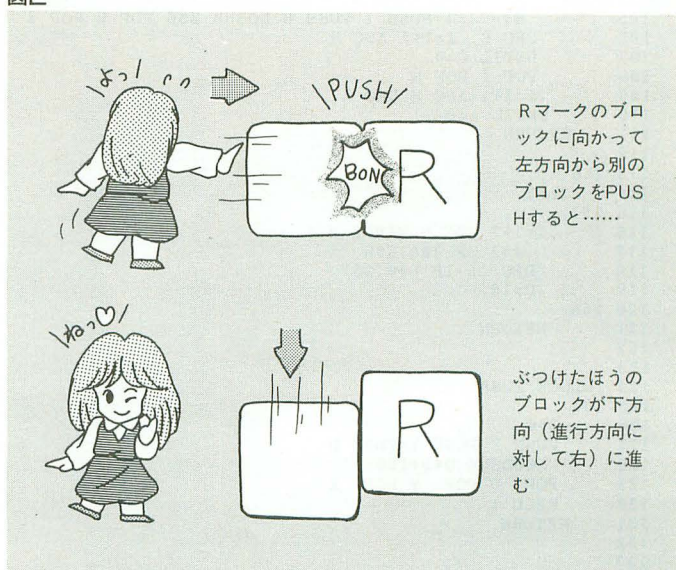



図2

せっかく作りあげたゲームを誰も見向きもしてくれないなんて悲しいですからね。

最後にTTCを使ったときに煩わしかったIF文の嵐も@～文のおかげですっきりしました。

初めのうちはTTCとの完全コンパチをなくしてまで命令の拡張を行う必要があるのかな？　と思っていましたが，結局これらの命令に助けられることになってしまいました。となると，拡張命令をサポートしたTTC＋＋(インクリメント)がほしくなるんだよなあ。誰か作ってくれません？

リスト1 PUSH BON!

```
  1 ;
  2 ; PUSH BON  FOR TTI
  3 ;    1989.9.9 BY J.YAMADA
  4 ;
  5
  6   'C' WIDCH 40
  7
  8 ;MAIN ROUTINE
  9    .R=0          ;ROUND 10 PRACTICE MEN
 10 9
 11    GOSUB 975 .G=5  GOSUB 980
 12       GOSUB 965  GOSUB 970
 13 10
 14    LOCATE  25,15 "PUSH ANY KEY!!"
 15       LOCATE  25,15 "              "
 16      IF (G=0,10
 17 11
 18       .T=0 .X=0 .Y=0 .D=0 .S=0
 19        GOSUB 980  BELL 1
 20 12
 21       .B=0 GOSUB 900 GOSUB 953 GOSUB 993
 22       .B=6 GOSUB 953
 23 13
 24       .K=(G IF K=0,13
 25         IF K='4,455
 26         IF K='6,456
 27         IF K='8,458
 28         IF K='2,457
 29         IF K='G,20
 30         IF K#' ,13
 31         GOSUB 300 INC S
 32         .A=0 ADC A IF A#0,20
 33         GOSUB 899
 34         GOSUB 250 IF O#0,23
 35         GOSUB 255 IF O#0,23
 36 14
 37        GOSUB 1000
 38      GOTO 13
 39
 40
 41 ;GIVE UP!!!
 42 20
 43    LOCATE 6,11 "             "
 44    LOCATE 6,12 "  GIVE UP!!  "
 45    LOCATE 6,13 "             "
 46      DEC  G GOSUB 980 BELL 2
 47 25
 48      GOSUB 990 IF G=0,21
 49     GOSUB 965 GOSUB 970
 50    GOTO 11
 51
 52
 53 ;GAME OVER
 54 21
 55    LOCATE 6,11 "* * * * * * *"
 56    LOCATE 6,12 " *GAME OVER* "
 57    LOCATE 6,13 "* * * * * * *"
 58    BELL 3
 59 22
 60     IF (G#' ,22
 61    .R=0 GOTO 9
 62
 63
 64 ;ROUND CLEAR
 65 23
 66    LOCATE 6,11 "* * * * * * *"
 67    LOCATE 6,12 " ROUND CLEAR "
 68    LOCATE 6,13 "* * * * * * *"
 69    BELL 3
 70 24
 71     IF (G#' ,24
 72     INC R IF R<10,25
 73     .R=0 GOTO 25
 74
 75
 76 ;* BLOCK YOKO CHECK
 77 250
 78    .H=$9F .I=00  .O=00
 79    .A=7 REPEAT
 80      .C=5 REPEAT
 81        WIND2 H,I .J=]
 82        @IF J=4 PUSH I PUSH H GOSUB 251 POP H POP I
 83       DEC C .I=I+1 ADC H
 84       UNTIL C=0
 85      .I=I+2 ADC H DEC A
 86     UNTIL A=0
 87    RETURN
 88
 89 251
 90    .L=2
 91 252
 92     .I=I+1 ADC H WIND2 H,I
 93       .J=] IF J#4,254
 94       DEC  L IF L#0,252
 95      .O=10
 96 254
 97      RETURN
 98
 99 ;* BLOCK TATE CHECK
100 255
101    .H=$9F .I=00  .O=00
102    .A=7 REPEAT  PUSH H PUSH I
103      .C=5 REPEAT
104        WIND2 H,I .J=]
```

```
105        @IF J=4 PUSH I PUSH H GOSUB 256 POP H POP I
106       DEC C .I=I+7 ADC H
107       UNTIL C=0
108       POP I POP H
109      .I=I+1 ADC H DEC A
110     UNTIL A=0
111   RETURN
112
113 256
114    .L=2
115 257
116    .I=I+7 ADC H WIND2 H,I
117     .J=] IF J#4,258
118     DEC  L IF L#0,257
119     .O=10
120 258
121     RETURN
122
123
124 ;BLOCK PUSH
125 300
126    .T=0
127    PUSH X PUSH Y PUSH D
128      @GOSUB D*5+150
129    POP  D POP  Y POP  X
130    BELL 1
131   RETURN
132
133
134 ;BLOCK UP PUSH
135 150
136   @IF Y<2 RETURN
137     DEC Y GOSUB 952 .J=B
138     @IF J=5 RETURN
139     @IF J=0 DEC S RETURN
140 151
141     .Q=Y DEC Y  IF Q<1,352
142       GOSUB 952 IF B#0,162
143       GOSUB 390
144     GOTO 151
145 352
146     .Y=Q .B=J GOSUB 953
147    RETURN
148
149
150 ;BLOCK RIGHT PUSH
151 155
152   @IF X>4 RETURN
153     INC X GOSUB 952 .J=B
154     @IF J=5 RETURN
155     @IF J=0 DEC S RETURN
156 156
157     .P=X INC X IF X>6,357
158       GOSUB 952 IF B#0,167
159       GOSUB 391
160       GOTO 156
161 357
162     .X=P .B=J GOSUB 953
163    RETURN
164
165
166 ;BLOCK DOWN PUSH
167 160
168   @IF Y>4 RETURN
169     INC Y GOSUB 952 .J=B
170     @IF J=5 RETURN
171     @IF J=0 DEC S RETURN
172 161
173     .Q=Y INC Y IF Y>6,352
174       GOSUB 952 IF B#0,162
175       GOSUB 390
176     GOTO 161
177 162
178     .W=Y .V=X .U=B .Y=Q .B=J
179    GOSUB 953 GOTO 370
180
181
182 ;BLOCK LEFT PUSH
183 165
184   @IF X<2 RETURN
185     DEC X GOSUB 952 .J=B
186     @IF J=5 RETURN
187     @IF J=0 DEC S RETURN
188 166
189     .P=X DEC X IF P<1,357
190       GOSUB 952 IF B#0,167
191       GOSUB 391
192       GOTO 166
193 167
194     .W=Y .V=X .U=B .X=P .B=J
195    GOSUB 953 GOTO 370
196
197
198 ;BLOCK WO HAJIKU
199 370
200    @IF T>10 .T=0 RETURN
201     INC T
202     @IF U>4 RETURN
203      IF J=2,371
204      IF J=3,376
205    IF U=2,374
206    IF U=3,377
207    RETURN
208
209
210 ;LEFT BLOCK
211 371
212    GOSUB 385
213 372
214       .J=U .X=V .Y=W
215      @GOTO D*5+151
216 374
217    GOSUB  385 @GOTO D*5+151
218
219
220 ;RIGHT BLOCK
221 376
222     GOSUB 386 GOTO 372
223 377
224     GOSUB 386 @GOTO D*5+151
225
226
227 ;DIR LEFT TURN
228 385
229    DEC D @IF D>121 .D=3
230    RETURN
231
232
233 ;DIR RIGHT TURN
234 386
235    INC D @IF D=4 .D=0
236    RETURN
237
238
239 ;BLOCK ERASE MOVE
240 390
241     PUSH Y .B=0 .Y=Q GOSUB 900 .B=0 GOSUB 953
242       POP  Y .B=J GOSUB 900
243     RETURN
244 391
245     PUSH X .B=0 .X=P GOSUB 900 .B=0 GOSUB 953
246       POP  X .B=J GOSUB 900
247     RETURN
248
249
250 ;CURSOR LEFT
251 455
252    .D=3  IF X=0,12
253      PUSH X .X=X-1 GOSUB 952
254       @IF B#0 POP X GOTO 12
255 454
256      .Q=X POP X  .B=0
257        GOSUB 900 .B=0 GOSUB 953
258       .X=Q GOSUB 993 .B=6 GOSUB 953
259       GOTO 14
260
261
262 ;CURSOR RIGHT
263 456
264    .D=1  IF X=6,12
265      PUSH X .X=X+1 GOSUB 952
266       IF B=0,454
267      POP  X
268    GOTO 12
269
270
271 ;CURSOR DOWN
272 457
273    .D=2  IF Y=6,12
274      PUSH Y .Y=Y+1 GOSUB 952
275       @IF B#0 POP Y GOTO 12
276 453
277       .Q=Y POP Y  .B=0
278        GOSUB 900 .B=0 GOSUB 953
279      .Y=Q GOSUB 993 .B=6 GOSUB 953
280     GOTO 14
281
282
283 ;CURSOR UP
284 458
285   .D=0  IF Y=0,12
286     PUSH Y .Y=Y-1 GOSUB 952
287      IF B=0,453
288     POP  Y
289    GOTO 12
290
291
292 ;BLOCK PRINT
```

▶友人Tは，教室の床においてあったビンに入った薄緑色の水彩絵の具を飲んでこういった。「これまじーよ」。　佐伯　章（16）東京都

```
293 900
294     .A=X*3+2 .C=Y*3+2 LOCATE A,C
295      @IF B>5 .B=0
296      IF B#5,902
297         "."      CHR $7B "."      'DLLL'
298         CHR $7B CHR $7B CHR $7B 'DLLL'
299         "."      CHR $7B "."
300      RETURN
301 902
302      IF B#0,901
303         "   " 'DLLL' "   " 'DLLL' "   "
304         RETURN
305 901
306      "+-+" 'DLLL' "I I" 'DLLL' "+-+"
307          LOCATE A+1,C+1
308          @GOTO  B+59
309     RETURN
310 60
311     " " RETURN
312 61
313     "L" RETURN
314 62
315     "R" RETURN
316 63
317     "*" RETURN
318
319
320 ;KVRAM ADRESS
321 950
322      .A=$9F .C=00 .F=Y
323        .C=C+X ADC A
324         IF F=0,951
325         REPEAT
326            .C=C+7 ADC A
327            DEC F
328         UNTIL F=0
329 951
330      WIND2 A,C
331     RETURN
332
333
334 ;KVRAM READ
335 952
336      GOSUB 950 .B=] RETURN
337
338
339 ;KVRAM WRITE
340 953
341      GOSUB 950 .]=B RETURN
342
343
344 ;MEN ADRESS
345 960
346      .H=$A0 .I=00  IF R=0,961
347       .D=R
348       REPEAT
349          .I=I+49 ADC H
350         DEC D
351       UNTIL D=0
352 961
353     RETURN
354
355
356 ;MEN TENSOU  READ
357 965
358     GOSUB  960  .A=$9F .C=00
359       .F=50 REPEAT
360         WIND2 H,I .B=]
361         WIND2 A,C .]=B
362         .I=I+1 ADC H .C=C+1 ADC A
363         DEC F
364       UNTIL F=0
365     RETURN
366
367
368 ;MEN PRINT
369 970
370       .Y=0  REPEAT
371        .X=0  REPEAT
372         GOSUB  950 .B=]
373         GOSUB  900
374            INC X   UNTIL X=7
375          INC Y   UNTIL Y=7
376      RETURN
377
378
379 ;WAKU PRINT
380 975
381      LOCATE 1,1  .A=23
382       REPEAT  "*" DEC A
383        UNTIL A=0
384          .Y=2 REPEAT
385           LOCATE  1,Y "*"
386           .A=21 REPEAT " " DEC A
387           UNTIL A=0  "*"
388           INC Y UNTIL Y=23
389         LOCATE 1,23  .A=23
390       REPEAT "*" DEC  A
391      UNTIL A=0
392      RETURN
393
394
395 ;SCREEN PRINT
396 980
397      GOSUB 990
398      GOSUB 899
399      GOSUB 992
400      GOSUB 991
401 RETURN
402
403
404 ;TITLE PRINT
405 990
406      LOCATE 25,2 " @@  @@@  @@ "
407      LOCATE 25,3 "@            @
408      LOCATE 25,4 "@ PUSH BON! @
409      LOCATE 25,5 "@            @
410      LOCATE 25,6 " @@  @@@  @@
411      LOCATE 25,21 "FOR TTI"
412      LOCATE 27,23 "BY J.YAMADA"
413      RETURN
414
415
416 ;STEP PRINT
417 899
418      LOCATE 26,8  "STEP  " PRT1 S
419      RETURN
420
421
422 ;ROUND PRINT
423 991
424      LOCATE 26,12 "ROUND " PRT1 R
425      RETURN
426
427
428 ;LIFE PRINT
429 992
430      LOCATE 26,10 "LIFE  " PRT1 G
431      RETURN
432
433
434 ;DIRECTION PRINT
435 993
436      LOCATE X*3+2,Y*3+2
437       @GOSUB 50+D
438      RETURN
439 50
440       'R' "^" 'DL' "I" 'DL' "I"
441       RETURN
442 51
443       'D' "-->" RETURN
444 52
445       'R' "I" 'DL' "I" 'DL' "V"
446      RETURN
447 53
448       'D' "<--" RETURN
449
450
451 ;WAIT
452 1000
453      .W=100 REPEAT .W=W-1 UNTIL W=0
454      RETURN
```

THE SENTINEL

リスト2 面データ

```
A000 00 00 00 00 00 00 00 00 : 00
A008 03 00 01 00 04 00 00 00 : 08
A010 04 02 01 03 05 00 05 00 : 14
A018 00 05 00 00 00 00 02 00 : 07
A020 00 00 00 00 00 00 01 04 : 05
A028 00 01 00 03 00 00 00 00 : 04
A030 00 00 00 05 00 00 00 00 : 05
A038 04 00 02 00 01 00 04 00 : 0B
A040 00 01 00 01 00 00 00 01 : 03
A048 03 01 03 01 00 00 00 02 : 0A
A050 01 02 00 00 00 00 05 04 : 0C
A058 05 00 00 00 00 00 00 00 : 05
A060 00 02 00 00 00 00 00 00 : 02
A068 00 00 01 05 04 00 03 00 : 0D
A070 00 00 01 04 00 01 00 00 : 06
A078 03 01 00 05 00 00 00 00 : 09
---------------------------------
SUM: 17 0F 09 1B 0E 01 14 0B BBE0

A080 02 00 01 00 04 00 05 00 : 0C
A088 03 00 00 00 00 00 00 00 : 03
```

▶僕は二十世紀梨ドリンクが好きです。駅にいったらよく買って飲みます。
森 弘 (21) 山口県

```
A090 00 02 00 00 00 05 00 00 : 07
A098 00 00 05 00 00 00 04 00 : 09
A0A0 03 04 00 02 00 01 05 00 : 0F
A0A8 00 00 01 01 00 00 00 00 : 02
A0B0 05 00 04 03 01 00 00 00 : 0D
A0B8 00 00 00 00 05 03 01 00 : 09
A0C0 02 00 00 00 00 00 00 04 : 06
A0C8 00 00 00 00 05 00 00 00 : 05
A0D0 03 00 00 00 00 05 03 00 : 0B
A0D8 00 03 00 01 04 01 00 02 : 0B
A0E0 01 00 02 05 00 00 00 00 : 08
A0E8 03 00 00 00 05 00 00 01 : 09
A0F0 00 04 00 00 00 00 04 00 : 08
A0F8 00 00 00 00 00 01 02 00 : 03
------------------------------------
SUM: 16 0D 0D 0C 18 10 18 07 0257

A100 00 03 00 00 00 01 05 04 : 0D
A108 00 00 00 03 05 05 05 02 : 14
A110 00 00 03 04 05 00 03 00 : 0F
A118 00 02 00 01 00 01 00 03 : 07
```

```
A120 00 00 01 00 00 00 00 00 : 01
A128 03 00 00 00 00 00 00 04 : 07
A130 05 00 03 05 00 01 03 04 : 15
A138 02 00 00 00 00 00 00 00 : 02
A140 01 00 00 01 02 05 01 00 : 0A
A148 00 00 00 00 01 02 00 04 : 07
A150 05 00 00 00 00 00 00 00 : 05
A158 00 00 00 00 04 05 00 03 : 0C
A160 00 00 01 00 00 00 01 00 02 : 04
A168 01 03 00 00 01 00 01 04 : 0A
A170 01 00 00 00 00 03 01 02 : 07
A178 00 01 00 00 03 00 02 00 : 06
------------------------------------
SUM: 12 0A 07 0E 15 18 15 20 4B66

A180 00 05 00 00 00 00 00 04 : 09
A188 00 00 00 01 00 00 00 00 : 01
A190 03 00 01 00 03 00 00 00 : 07
A198 02 00 02 00 00 01 05 01 : 0B
A1A0 04 01 05 04 00 00 02 01 : 11
A1A8 02 00 00 00 03 00 01 00 : 06
```

```
A1B0 03 00 00 00 00 00 04 00 : 07
A1B8 00 00 00 00 00 00 00 00 : 00
A1C0 00 03 02 02 02 02 00 00 : 0B
A1C8 00 05 04 05 00 00 00 03 : 11
A1D0 04 05 04 03 00 00 05 02 : 17
A1D8 00 03 05 00 00 00 03 03 : 0E
A1E0 02 00 00 00 00 00 03 00 : 05
A1E8 00 00 00 00 00 00 00 00 : 00
A1F0 00 00 01 00 02 00 01 00 : 04
A1F8 00 00 00 00 00 00 00 00 : 00
------------------------------------
SUM: 14 16 18 0F 0A 03 18 0E 29EB

A200 00 02 02 00 03 00 00 03 : 0A
A208 03 02 00 05 00 00 00 03 : 0D
A210 03 00 00 00 00 00 00 04 : 07
A218 00 04 04 CD 17 9F C1 04 : 50
------------------------------------
SUM: 06 08 06 D2 1A 9F C1 0E C958
```

全機種共通システムインデックス

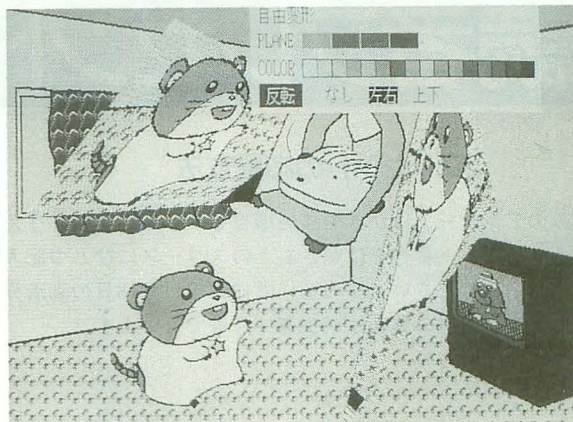
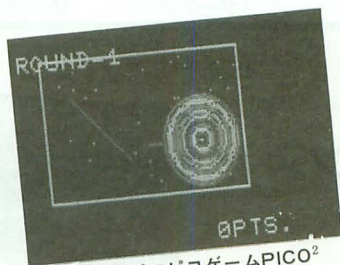
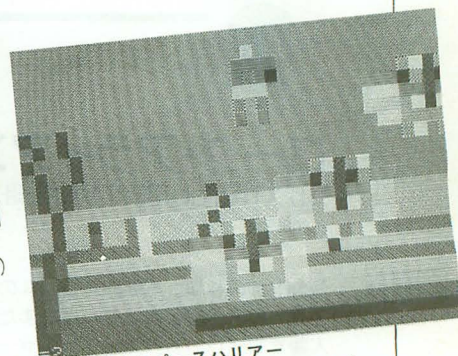
■85年6月号
序論　共通化の試み
第1部　S-OS"MACE"
第2部　Lisp-85インタプリタ
第3部　チェックサムプログラム
■85年7月号
第4部　マシン語プログラム開発入門
第5部　エディタアセンブラZEDA
第6部　デバッグツールZAID
■85年8月号
第7部　ゲーム開発パッケージBEMS
第8部　ソースジェネレータZING
■85年9月号
インタラプト　S-OS番外地
第9部　マシン語入力ツールMACINTO-S
第10部　Lisp-85入門(1)
■85年10月号
第11部　仮想マシンCAP-X85
連載　Lisp-85入門(2)
■85年11月号
連載　Lisp-85入門(3)
■85年12月号
第12部　Prolog-85発表
■86年1月号
第13部　リロケータブルのお話
第14部　FM音源サウンドエディタ
■86年2月号
第15部　S-OS"SWORD"
第16部　Prolog-85入門(1)
■86年3月号
第17部　magiFORTH発表
連載　Prolog-85入門(2)
■86年4月号
第18部　思考ゲームJEWEL
第19部　LIFE GAME
連載　基礎からのmagiFORTH
連載　Prolog-85入門(3)
■86年5月号
第20部　スクリーンエディタE-MATE
連載　実戦演習magiFORTH
■86年6月号
第21部　Z80TRACER
第22部　magiFORTH TRACER
第23部　ディスクダンプ&エディタ
第24部　"SWORD" 2000 QD
連載　対話で学ぶ magiFORTH
特別付録　PC-8801版S-OS"SWORD"
■86年7月号
第25部　FM音源ミュージックシステム
付録　FM音源ボードの製作
連載　計算力アップのmagiFORTH
特別付録　SMC-777版S-OS"SWORD"
■86年8月号
第26部　対局五目並べ
第27部　MZ-2500版S-OS"SWORD"
■86年9月号
第28部　FuzzyBASIC発表
連載　明日に向かって magiFORTH
■86年10月号
第29部　ちょっと便利な拡張プログラム
第30部　ディスクモニタ DREAM
第31部　FuzzyBASIC 料理法〈1〉
■86年11月号
第32部　パズルゲーム HOTTAN
第33部　MAZE in MAZE
連載　FuzzyBASIC 料理法〈2〉
■86年12月号
第34部　CASL & COMET
連載　FuzzyBASIC 料理法〈3〉
■87年1月号
第35部　マシン語入力ツールMACINTO-C
連載　FuzzyBASIC 料理法〈4〉
■87年2月号
第36部　アドベンチャーゲーム MARMALADE
第37部　テキアベ作成ツール CONTEX
■87年3月号
第38部　魔法使いはアニメがお好き
第39部　アニメーションツール MAGE
付録　"SWORD" 再掲載と MAGIC の標準化
■87年4月号
第40部　INVADER GAME
第41部　TANGERINE
■87年5月号
第42部　S-OS"SWORD" 変身セット
第43部　MZ-700用 "SWORD" を QD 対応に
■87年6月号
インタラプト　コンパイラ物語
第44部　FuzzyBASIC コンパイラ
第45部　エディタアセンブラ ZEDA-3
■87年7月号
第46部　STORY MASTER
■87年8月号
第47部　パズルゲーム碁石拾い
第48部　漢字出力パッケージ JACKWRITE
特別付録　FM-7/77版 S-OS"SWORD"
■87年9月号
第49部　リロケータブル逆アセンブラ Inside-R
特別付録　PC-8001/8801 版 S-OS"SWORD"
■87年10月号
第50部　tiny CORE WARS
第51部　FuzzyBASIC コンパイラの拡張
第52部　X1turbo 版 S-OS"SWORD"
■87年11月号
序論　神話のなかのマイクロコンピュータ
付録　S-OS の仲間たち
第53部　もうひとつの FuzzyBASIC 入門
第54部　ファイルアロケータ&ローダ
インタラプト　S-OS こちら集中治療室
第55部　BACK GAMMON
■87年12月号
第56部　タートルグラフィックパッケージTURTLE
第57部　X1turbo 版 "SWORD" アフターケア
　　　　ラインプリントルーチン
特別付録　PASOPIA7 版 S-OS"SWORD"
■88年1月号
第58部　FuzzyBASIC コンパイラ・奥村版
付録　石上版コンパイラ拡張部の修正
■88年2月号
第59部　シューティングゲーム ELFES
■88年3月号
第60部　構造型コンパイラ言語 SLANG
■88年4月号
第61部　デバッギングツール TRADE
第62部　シミュレーションウォーゲーム WALRUS
■88年5月号
第63部　シューティングゲーム ELFES II
第64部　地底最大の作戦
■88年6月号
第65部　構造化言語 SLANG 入門(1)
第66部　Lisp-85 用 NAMPA シミュレーション
■88年7月号
第67部　マルチウィンドウドライバ MW-1
連載　構造化言語 SLANG 入門(2)
■88年8月号
第68部　マルチウィンドウエディタ WINER
■88年9月号
第69部　超小型エディタ TED-750
第70部　アフターケア WINER の拡張
■88年10月号
第71部　SLANG 用ファイル入出力ライブラリ
第72部　シューティングゲーム MANKAI
■88年11月号
第73部　シューティングゲーム ELFES IV
■88年12月号
第74部　ソースジェネレータ SOURCERY
■89年1月号
第75部　パズルゲーム LAST ONE
第76部　ブロックゲーム FLICK
■89年2月号
第77部　高速エディタアセンブラ REDA
特別付録　X1版 S-OS"SWORD"〈再掲載〉
■89年3月号
第78部　Z80用浮動小数点演算パッケージSOROBAN
■89年4月号
第79部　SLANG 用実数演算ライブラリ
■89年5月号
第80部　ソースジェネレータ RING
■89年6月号
第81部　超小型コンパイラTTC
■89年7月号
第82部　TTC用パズルゲーム　TICBAN
■89年8月号
第83部　CP/M用ファイルコンバータ
■89年9月号
第84部　生物進化シミュレーションBUGS
■89年10月号
第85部　小型インタプリタ言語TTI

＊以上のアプリケーションは，基本システムである S-OS "MACE" または S-OS"SWORD" がないと動作しませんのでご注意ください。

投稿プログラム大募集
のお知らせ

Oh!Xでは，毎月さまざまな投稿プログラムを掲載しております。これらはすべて，ゲーム音楽を聞いているうちに自分のマシンで演奏してみたくなった，市販のものもあるけどもっと便利なグラフィックツールが欲しかった，またはMZ-700でスペースハリアーを遊びたいなど，どれも皆さんが日常のなかでパソコンと接しているうちに，ふと思いついたことを形にしようと努力して生み出された傑作，名作ばかりなのです。

でも，読者の皆さんがそうして作り上げたプログラムを，一部の方を除いては自分のディスクのなかだけにしまっておくのはもったいない話。ひとりでも多くのユーザーに使ってもらえば，またそれをベースにして新しいプログラムが生まれる可能性だって広がるのです。

ですから，Oh!Xではそういったちょっとしたきっかけを機に，完成度の高いものよりも自分のアイデアをそのまま形にしたような，オリジナリティあふれる投稿プログラムをスペースを空けてお待ちしています。もちろん，ピコピコゲームのようなショートプログラムも大歓迎。自信作をお持ちの方は，募集要項をよくお読みのうえぜひご参加ください。お待ちしています。

MZ-2500用グラフィックツールDMACS（1988年9月号）

MZ-2500用ピコピコゲームPICO²（1988年4月号）

MZ-700用スペースハリアー（1988年10月号）

X1/X1 turbo用割り込みミュージックシステムPSI（1988年3月号）

X68000用ストラテジーゲームSTAR TREK（1988年11月号）

S-OS"SWORD"用ELFES（1988年2月号）

X1turbo用レイトレーシングツールturbo RAY TRACER（1988年9月号）

投稿募集要項

1）お送りいただくプログラムには，住所・氏名・年齢・職業・連絡先電話番号・機種名・使用言語・必要な周辺機器・マイコン歴等を明記のうえ，封書の宛て先の最後には「Oh!X LIVE」や「S-OS"SWORD"」,「投稿ゲームプログラム」など，プログラムの内容を明確にご記入ください。

2）投稿されるプログラムには，詳しい内容を記入した原稿と一緒にフローチャート，変数表，メモリマップ，参考文献などの資料もお書き添えのうえお送りください。また，お送りいただいた原稿については，当方で加筆，修正させていただく場合があります。

3）お送りいただくプログラムは最低2回はセーブしてください。基本的に同封されたカセットテープおよびフロッピーディスクについてはご返送いたしませんので，あらかじめご了承ください。

4）ハード製作関係の投稿につきましては，最初は詳しい内容のわかる原稿のみお送りいただければ結構です。その後，当方において製作物が必要だと判断した場合は，改めてご連絡いたします。

5）お送りいただいた投稿プログラムの採用につきましては，掲載月号が決定した時点で当方よりご連絡を差し上げます。特に各種ツール関係，ハード関係のものにつきましては，特集内容などを考慮したうえで採用が決定されることがありますので，採用結果をご連絡するまでに時間がかかってしまう場合もあります。

6）投稿いただいたプログラムにバグ等が発見された場合には，新しいプログラムの入ったメディアと一緒に，文書にてご連絡ください。

7）掲載された投稿プログラムに対しては当社規定の原稿料をお支払いたします。また，プログラムの著作権等は制作された方に保留されますが，PDSとしてネットなどにアップロードされる場合は，必ず編集室まで事前にご連絡ください。なお，一般的モラルとして，他誌との二重投稿または，他誌に掲載されたプログラムの移植などについては固くお断わりいたします。

宛て先
〒102　東京都千代田区九段南2-3-26　井関ビル
日本ソフトバンク　Oh!X編集室「投稿プログラム」係

愛読者プレゼント

プレゼントの応募方法

とじ込みのアンケートはがきの該当項目をすべてご記入のうえ，希望するプレゼント番号をはがき右下のスペースにひとつ記入してお申し込みください。締め切りは1989年11月18日の到着分までとします。当選の発表は1990年1月号で行います。

1

ホビージャパン　☎03(354)9341

リングマスターI

X68000用5"2HD版3枚組　8,800円

テーブルトーク感覚のRPG。時代背景がしっかりしているので，ゲームの世界におもいっきりはまれるというのがミソ。キーボードからの会話入力がたのしい。

3名

2

スタークラフト　☎03(988)2988

3名

ローグ・アライアンス

X1/X1turbo用5"2D版3枚組
9,800円

海外ものの移植。比較的バランスの良いゲームなので，RPG初心者でも，はたまたRPG大好き人間でも楽しめる。アドベンチャーモードも付いている。

3

サン・ミュージカル・サービス　☎03(419)8839

ソングファイル 68Kシリーズ

Musicstudio PRO-68Kシリーズ対応
A.佐藤允彦/リゾーム症候群
B.関根安里/スケッチ
X68000用5"2HD版　各5,800円
各2名

Musicstudio PRO-68K用のオリジナルデータ曲集。ローランドのMT-32に音色を対応させている。アレンジメントが自宅で手軽に楽しめるスグレもの。

4

アンス・コンサルタンツ　☎092(522)6347

サイクロンオリジナルTシャツ

5名

アンス・コンサルタンツのサイクロンオリジナルTシャツをプレゼント。色は，赤，青，黄，緑，白の5色。各1名ずつ，計5名に差し上げます。

5

コーヒーキャンディ

1名

満開製作所からの配給物。祝氏が北海道は女満別にて入手したコーヒーキャンディだ。"珈琲中毒"というネーミングと，パッケージのすばらしさには感動せざるをえない。意外にも味は良い。

9月号プレゼント当選者

[1]サイクロンExpress（長野県）土屋和幸　[2]ジェノサイド（東京都）大西伸一　池田和繁（神奈川県）石井典雄　[3]琉球（滋賀県）野村慎一郎（宮城県）西山一法（石川県）久下沼信　[4]ソーサリアン・スーパーアレンジバージョン（千葉県）渡瀬慎一郎（奈良県）森芳生（広島県）木下研一　[5]コンピュータ・アニメーション　Vol. 1（山梨県）斎藤真二（島根県）坂本真治　Vol. 2（東京都）町田富士男（大阪府）岸雅樹　（敬称略）

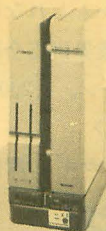
以上の方々が当選されました。おめでとうございます。商品は順次発送いたしますが，入荷状況などにより遅れる場合もあります。また，公正取引委員会の告示により，このプレゼントに当選された方は，この号の他の懸賞には当選できない場合がありますので，ご了承ください。

（価格はすべて消費税別です）

PENGUIN INFORMATION CORNER

ペ・ン・ギ・ン・情・報・コ・ー・ナ・ー

NEW PRODUCTS

X68000用ハードディスクドライブ
HXD040/HXD042
アイテム

アイテムは，X68000専用に開発した40Mバイトのハードディスクを販売開始した。従来のX68000用の汎用ディスク装置とは異なり本体の下にそのまま設置可能なサイズとなっており場所をとらない。最大80Mバイトまでのディスクが利用できる。

OSは，Human68kの全バージョンとOS-9の両方に対応しており，分割フォーマットも可能。フォーマット時に不良セクタの代替処理を行うことができるほか，切電時のオートパーキングロック機能も持っている。平均アクセス速度は23msとこのクラスではもっとも速いものとなっている。

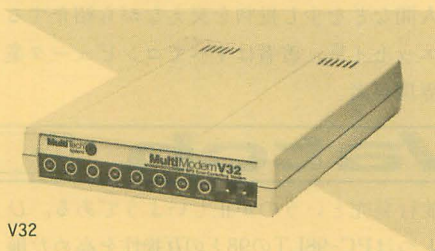
価格は,1台目用モデルのHXD040が118,000円,2台目用モデルのHXD042が128,000円となっている。HXD042はX68000ACE-HD, EXPERT-HDなどハードディスク内蔵モデルに接続することも可能。

〈問い合わせ先〉
(株)アイテム　☎03(434)4171

X68000用周辺ボード
FREELANCE BOARDシリーズ
八戸ファームウェアシステム

X68000用の周辺ボード群が発売される。発売されるのは，A/D変換ボード(12/13ビット)，D/A変換ボード(同)，48ビットのPIOボード，GP-IBボード，24点絶縁型入力ボード，24点絶縁型出力ボード，リードリレーボード，25ビットのUp/downカウンタボードなど。発売は10月の予定で価格は未定である。同時に，X68000 2台で1台のプリンタを共有する手動式のプリンタ切り替え機も販売する。こちらは19,800円となっている。

〈問い合わせ先〉
八戸ファームウェアシステム(株)
☎011(716)3815

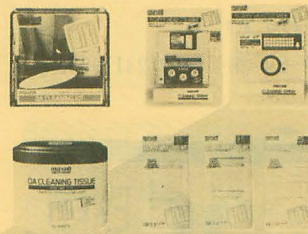
CCITT V・42対応のモデム2製品
Multi Modem V32/224EHI-5
コア

V32

コアは，日本で初めてのCCITTV.42規格に対応した全二重モデムを発売した。V.42はCCITTで1988年4月に採択されたモデムエラー訂正方式のプロトコルで，ISDNのDチャネル用プロトコルLAP-MとMNPクラス3/4の両方の機能を持つ。

販売開始されたのは，非同期式での通信速度が19200bps（同期式は半分）で360,000円のMulti Modem V32と4800bps(同じく)で110,000円のMulti Modem 224EH-1の2製品。両製品とも，データ圧縮方式にはMNPクラス5を採用，制御手順はヘイズAT準拠となっている。

〈問い合わせ先〉
(株)コア　☎045(441)8611

パソコン用アクセサリーキット
WiKiシリーズ
日立マクセル

日立マクセルは，ワープロ・パソコン用のアクセサリーWiKiシリーズの販売を開始した。販売開始されたものは，静電気除去

WiKiシリーズ

ブラシやクリーニングスプレーなどから構成されるOAクリーニングキット(2,800円)，湿式のOAクリーニングティシュ(1,000円)，5インチ/3.5インチの湿式フロッピーヘッドクリーナー(2,500円)，フロッピーインデックスラベルキット(300円)など。

〈問い合わせ先〉
日立マクセル(株)　☎03(241)0736

インクジェットプリンタ
IO-735X
シャープ

シャープは,新型カラーインクジェットプリンタの販売を開始した。同製品は，すでに販売しているIO-735の後継機種であり，新たにESC/Pコマンドに対応した。使用可能な機種は，MZ，X1，X68000，AX，PC-9801などのパソコンとWD-910など。

印刷は専用紙以外にも普通紙，葉書，OHPシートなどへモノクロまたは7色カラーで印刷することができる。用紙サイズは最大B4。出力は1色あたり12ノズル計48ノズルのマルチノズル方式を採用したため，インク交換は1色単位のカートリッジの交換だけでよい。価格は248,000円。

〈問い合わせ先〉
シャープシステムプロダクト(株)　プリンタ事業部　☎03(459)8842

セキュリティ機能に優れた
ICカード関連製品
日立マクセル

日立マクセルは，今秋からICメモリカードなどの販売を開始する。販売が開始されるのは，8KバイトのRAMとプログラム領域

10Kバイトの MPUを1チップ化したICカード，バックアップ可能な512KバイトRAM搭載のメモリカードなど。特にICカードはISO規格に準拠するとともに暗号化処理，複数暗証番号，取り引き確認情報など高度なセキュリティ機能を持たせられ，専用の言語でパソコンからの制御することもできる。
〈問い合わせ先〉
日立マクセル(株)　☎03(241)9736

INFORMATION

マイクロウェア日本法人設立

米マイクロウェア・システムズ・コーポレーションは，日本法人であるマイクロウェア・システムズ(株)を設立すると同時にOS-9関連製品の販売を開始する。

従来，OS-9および関連製品の販売は，米マイクロウェアと(株)マイクロボードの合弁会社マイクロウェア・ジャパンが行ってきたが9月に合弁事業を中止し，日本法人を設立した。新会社の設立と同時にOS-9用アプリケーションソフトウェアの開発を行っていた(株)星光電子はマイクロウェア・システムズに吸収された。同社では今後，OS-9以外の新規事業も予定している。
〈問い合わせ先〉
マイクロウェア・システムズ(株)
☎03(839)9000

BOOKS

当然 パソコン事情ハンドブック
ラディク・ラボ

本書は，コンピュータに関係する会社，人間などを少し批判を交えながら紹介するエッセイ集。著者はすべてコンピュータ業界関係者。

副題に「心にはたらく業界ガイド」とあるように，今からコンピュータを始めようとしている人がコンピュータ業界の実情についての知識を得ることもできる。とはいっても，書いてあることに多少誇張があるのも事実なので，純然たる読み物として読んだほうがよいかもしれない。
コンピュータ業界研究会編著
B6判，207ページ，1,200円
〈問い合わせ先〉
(株)ラディク・ラボ　☎03(235)8061

幻夢年代記
ビジネスアスキー

本書は，1984年から1988年の間にLOGIN誌に連載されたものを単行本化したもの。

Again Watch

Short Again特集

先月号ではひとつの話題に終始してしまったので，今回はたくさんの話題を盛り込みたいと思う。名づけて「Short Again特集」。

パソコン業界では例年，秋の新製品発表会となるデータショウ，エレクトロニクスショウに合わせていろいろと新しいパソコンが披露される。ということはメーカーの戦略上この時期の新製品の発表は行われないという事実がある。よって，9月末になった現在，まだ目新しい新製品はメーカーから発表されていないということから今回は単にネタがないだけだったりということもいえるのだが。

まだ発表がないNECのブック98

おそらくこの本がみなさんのお手元に届く頃にはNECから発表がされていると思うが，今の時点では，まだブック98はリリースどころかアナウンスメントすらされていない。

いろいろと情報が飛び交っているのだが，総合すると，一部新聞報道にあった2機種並行発売というのが正しいようである。ひとつはPC-98LTの98との互換性を高めた廉価製品。もうひとつはPC-9801LV22の小型軽量低価格版。この2つを東芝「ダイナブック」(198,000円）の価格付近にバランスよく配置して対抗してくる模様だ。いまにして思えば，1986年に発売されたPC-98LTは238,000円という当時としては画期的な低価格だった。したがって，いまのコストに換算すると，150,000円でもおかしくはない。超低価格を売り物にしたダイナブックよりも安く作ることは十分可能というわけだ。

実際に，蓋を開けると何が飛び出してくるのか？　良くも悪くもNECは業界のリーダー。やるときにはやってほしい。最近はちょっと安易にお茶を濁した新製品が多すぎるような気がする。

386Vで32ビット機は変わるか？

セイコーエプソンはこれまで386マシン「PC-386」を598,000円で販売してきたが，ようやくこの廉価普及版「PC-386V」を投入した。拡張スロットの数を4ボードから2ボードに減らして価格をちょうど100,000円安くした。これはNECの「PC-9801RA2」と同じ価格である。

実際にはRA2は1年以上前の商品なので，エプソンはもっと安く設定してもよかった。ただし完全な互換性が保証されないためかあまり人気がないエプソン機の店頭値引き率はけっこう高いという現実がある。したがって，採用したCPUの性能(クロック周波数)が20MHzと高い(RA2は16MHz)ことを合わせて考えると，ほどよい価格に落ち着き，コストパフォーマンスとしては同じようなものだろう。

しかし，32ビット機はCPU自身，すなわちインテルの386販売価格が値下がりしている割には安くなっていない感じがする。ソニーのAXパソコン「クオーターL」は外部バス16ビットの386SXでありながら298,000円とひとり頑張っている印象を受ける。NECもおそらくこの秋に新しい386マシンを投入すると思うが，後継CPUである486も出たことだし，32ビット機がもっと身近になってくれてもいいと思う。その意味でエプソンPC-386Vの登場にはちょっと期待している。

題名からも推測されるように，著者は元々はSF文学の翻訳，紹介を手掛けており，これは，「パソコンゲーム年代期」のようなものである。連載が開始された年は世間ではオーウェルの「1984」がはやっていたが，パソコンゲーム元年みたいな年でもあったらしい。本書では，1984年の「ウルティマIII」から1988年の「アメリカ大統領選1988年版」までのゲームを紹介している。パソコンゲームの黎明期から現代までの歴史を年代順に追ってみたいという人には絶好の書。

安田均著

B6判，326ページ，1,800円

〈問い合わせ先〉

(株)ビジネス・アスキー　☎03(486)7119

X68000ノーライフキングに出演

X68000がスクリーンに登場する。この12月に劇場公開が予定されている『ノーライフキング』(市川準監督作品)がそうだ。原作は，マルチクリエイターいとうせいこうのベストセラーで，人気ソフト「ライフキング」をめぐる子供たちの「リアル」な世界を描いた物語である。本誌でも昨年12月号の「われら電脳遊戯民」で荻窪圭氏がこの『ノーライフキング』を取り上げ，新しい時代のキーワードとなる「リアル」について語っている。

映画では，舞台のひとつとなる学習塾に30台ものX68000が並べられるのをはじめ，全編を通してマンハッタンシェイプのツインタワーがスクリーンに映し出されるシーンが続出する。また，子供たちが遊ぶシューティングゲーム，そ

して，いわゆるファミコンゲームである「ライフキング」のグラフィック画面も実はX68000によって作成されている。メモリを大量に積んだX68000がハードディスクをガシガシやりながらスクリーン目一杯に生み出すアニメーションは圧巻だ。

制作にあたっては，LOGIN誌でお馴染みのスタッフも参加しているようである。X68000ファンとしては見逃せない作品だろう。

●

監督：市川　準
出演：高山良/鈴木さえ子/嶋田久作
提供：アルゴ・プロジェクト
制作：ニュー・センチュリー・プロデューサーズ／新潮社／サントリー株式会社
1989年度作品　ビスタサイズ
1時間46分　フジカラー

EISAは計画頓挫？

さて日電とエプソンの話題が出たところで唯一両社が共同戦線を進めているパソコン標準化プロジェクトがある。米国で32ビット幅の新PC/ATバスを作ろうという「EISA(イーサ)」計画だ。EISAは16ビットのPC/ATバスを互換性を保ったまま32ビットに移行しようという計画で，世界で最も普及しているPC/ATユーザーには歓迎された。一時期はもっぱらの話題になったが，さてその後は？

現実にインテルはEISAに対応したLSIチップセットの開発を終了した。しかしながら，メーカーとして採用，商品化を表明したところはひとつもない。

関係者によると，これはリーダー役であるコムパック・コンピュータ社がIBMのPC/ATの後継機種であるPS/2に採用された32ビットバス マイクロチャネル(MCA)に寝返り，EISAを進める気をなくしたというのが理由とされている。ほかのワング・ラボラトリ，ASTリサーチ，ゼニス・データ・システムズからも音なしの構えだし，計画は終わってはいないものの，極度にスローダウンしている，というさびしい状況であるようだ。

米国からようやくIBMのPS/2の出足が順調になってきているとの話が伝わってくるようになってきた。そうなると互換機最大手のコムパック・コンピュータとしては実際問題として後を追うしかないのは自明の理。とはいえ初の互換機メーカーが集まったプロジェクトだっただけに成功すれば面白かったのだが。互換機メーカーはしょせん，互換機メーカー，本家に楯突くことはできないということか。ちなみに，IBMはPS/2の最新機種として486搭載の製品の出荷を開始したという話も聞いている。

極度に不振のパソコン夏商戦

話は国内。夏のパソコン商戦は散々な成績だったようだ。2位のエプソンでさえも7～9月は6月以前に比べると絶不調に陥った模様で，大々的に広告を打ち上げたにもかかわらず富士通のFM-TOWNSなんかは誰が買っているの？　という状態だという話すら聞く。ところが奇妙なことにNECのPCシリーズだけは7～9月期も4～6月期比で台数ベース数割減に食い止めたという情報がある。理由は一切不明だが，現実に今年度上半期（'89年4～9月期）は予定の35万～40万台に収まる模様だという。

下半期は東芝のダイナブックが本格的に市場にリリースされるなど，NEC以外のパソコンの動きが活発化する要因もいくらかあるので期待できる。

メモリが安くなる

1MビットのダイナミックRAMがようやく1個1,800円から価格が急降下しはじめた。10月には1,600円になり，年末には確実に1,500円を割り込む見込みである。これによって，これまで高額商品となっていたEMSボードやRAMディスクが徐々に低価格化することは間違いない。

さらに，ハードディスクもガンガンと安くなっている。ついにアクセス速度28ms40Mバイト製品が10万円ちょっとで買える程の価格帯にまで落ちてきた。1年前は20Mバイトで15万円はしたのだが。恐ろしい話である。(K.T.)

FILES Oh!X

このインデックスは，タイトル，注記――筆者名，誌名，月号，ページで構成されています。台風も過ぎ去って，めっきり秋らしくなってきました。今が一番食べ物もおいしい季節，でも食べ過ぎにはご注意を。

参考文献
I/O　工学社
ASCII　アスキー
テクノポリス　徳間書店
bit　共立出版
POPCOM　小学館
マイコン　電波新聞社
マイコン BASIC Magazine　電波新聞社
LOGIN　アスキー

一般

▶柔らかい電楽　第4回
MIDIシステムをバンドに取り入れて，スタジオでメンバーとの同期演奏にチャレンジした体験記。――JOLL JOLL CLUB，マイコン，10月号，240-245pp.

▶次世代ヒューマンインタフェースへの期待
現世代のヒューマンインタフェイスをとおし，これからのコンピュータのあり方や期待を論じている。――田村浩一郎，bit，10月号，4-16pp.

▶ビジネスマンの情報管理術
通信ケーブルとフロッピーを使っての電子手帳のデータの保存と，電子手帳どうしのデータ交換の方法について解説。――塚田洋一，マイコン，10月号，316-320pp.

▶ブローダーバンド"電子手帳くん"新発売
ブロダーバンドから，PC-9801シリーズと電子手帳の間でデータ交換が可能になるインタフェイスが登場。その内容と評価をしている。――丹治佐一，マイコン，10月号，321-325pp.

▶ NeXT Computer System
NeXTの連載レポート，今回はソフトウェアのなかから，NextStepとMachについての概念を紹介。――編集部，ASCII，10月号，285-294pp.

▶特集ディスプレイの不思議
パソコン本体に比べると，あまり注目されないディスプレイ。けれどハードユーザーなら，ディスプレイにもこだわりたい。カラー液晶など最新技術から，原理，未来のディスプレイ，選び方などを詳しく解説，紹介。――編集部，LOGIN，18号，114-129pp.

▶図解世界のコンピュータちゃん　第20回
今回はナムコの疑似3D体感ゲーム，ウイニングランを紹介している。3Dグラフィックの陰影付けやポリゴンのメモリ上の展開などを解説。そのほか，コンピュータ用語独断的解説大事典も参考になる。――編集部，LOGIN，18号，160-161pp.

▶ネットワーカー・ホリック　第7回
パソコン通信を取り扱った後藤久美子主演のTBSのドラマ「空と海をこえて」を紹介。――編集部，LOGIN，18号，218-221pp.

▶電子音楽塾　第1回
MIDIって何だ？　と題して，まず言葉の意味，利点，コンピュータとの関連などについて説明している。――編集部，LOGIN，18号，229p.

▶ハードラボラトリー
パソコン用高品位プリンタ特集。エプソンやNECを始め，シャープのCZ-8PC4も紹介されている。――編集部，POPCOM，10月号，108-110pp.

▶ハイテク地獄耳
シャープのワープロ，書院WD-HL30，WD-A800/1800とCDラジカセQT-50CDを紹介。――編集部，POPCOM，10月号，128-133pp.

▶パソコン通信局を開局しよう！
パソコン通信ホストを個人で開局するためのノウハウを簡潔に紹介。電話回線やモデム，ホストプログラムについてや，開局に当たってのネットワークホスト設計など。――ITOCHI，マイコン BASIC Magazine，10月号，43-46pp.

MZ-80K/C/1200/700/1500

MZ-700/1500(S-BASIC)

▶イモムシの一生
4本立てのショートプログラムゲーム。メニューは孤独なイモムシ，風呂のタイルぬり，ランダムボール，テロリスBTS社。――村田達也，マイコン BASIC Magazine，10月号，128-129pp.

MZ-700/1500(HuBASIC)

▶ DESTROY
敵は暴走した電波軍の防衛コンピュータ。8方向に動く自機を操り敵を破壊しろ！――水谷邦久，マイコン BASIC Magazine，10月号，130-131pp.

MZ-1500

▶あしたのジョー
非リアルタイムボクシングゲーム。3人のライバルボクサーと闘う。――ヘルニャン・バウマン，マイコン BASIC Magazine，10月号，132-133pp.

MZ-1200

▶誌上公開質問状
MZ-1200用のプリンタと周辺機器を紹介。――編集部，マイコン BASIC Magazine，10月号，65-66pp.

MZ-80B/2000/2500/2800

MZ-2000/2500(1Z001)

▶ Hong Kong（香港）
移植版。パイを使った究極のパズルゲーム。――山下貴史，マイコン BASIC Magazine，10月号，134-136pp.

MZ-2500

▶酔いどれ天使
天使を操って，画面上の酒を全部ごみ箱に捨てるという，もったいないゲーム。――謎のパズル大好きおじさん，マイコン BASIC Magazine，10月号，137-139pp.

X1/X1turbo/Z

X1シリーズ

▶ 3 short games

いうまでもなく，スティーブ・ジョブズというのはApple社を興しApple-IIを売りMacを作らせ最近NeXTを発表したジョブズである。経歴を見てさぞや偉い人だろう，アイアコッカ風のビジネス書かな，と思うと後悔する。描かれているのは成功したビジネスマンではなく，ただの傍若無人な若僧そのものなのだ。本書でも創造性がまったくない，人の手柄を横取りしてしまうなど彼を普通の基準で見ると悪い面の方が多い。それでも一気に読めてしまうのは，ジョブズの一本気なパワーと当時のマニアたちが持つエネルギーが伝わってくるからだ。なんだかんだいってジョブズがApple社を追い出されるように辞めた後でもApple社は彼の業績から逃れていない。不振に喘いでいたMacの株を上げたレーザーライターはジョブズが周囲の反対を押し切って開発を進めていたものだし，最近発表されたポータブルマックも彼が以前から作ろうと頑張っていた製品なのだ。

ただ，本書は登場人物紹介を付けて貰いたいほどの入り組んだ人間関係と時間の流れが整理されていないので，そこだけが残念である。　（K）

スティーブ・ジョブズ　ジェフリー・S・ヤング著　日暮雅通訳　JICC出版　☎03(221)1997
A5判　346ページ（上）（下）各1,600円

書籍の価格は消費税込みです

ショートプログラム3本。「LONG HOLE」「BOUND BALL」「DEFENCE」。――横関薫，マイコンBASIC Magazine，10月号，165-166pp.

▶ Mr.COMBAT
ダイナマイトを連鎖爆発させて敵戦車を破壊する。パズルっぽいゲーム。――宮本進，マイコンBASIC Magazine，10月号，167-169pp.

X1+FM音源ボード（要NEW FM音源ドライバ）

▶サンダークロス　―エンディング―「A Long Way」
ゲームミュージックプログラム。――上田順一，マイコンBASIC Magazine，10月号，200-201pp.

X1turboシリーズ

▶くるくるパネル
パネルを引っくり返して敵を落とすゲーム。クインティがヒントだそうだ。――吉田紀生，マイコンBASIC Magazine，10月号，170-171pp.

▶ LET's PROGRAMING！
10進数をN（2～9）進数へ変換するプログラムの宿題発表。X68000用，X1turbo用など。――藤本健，マイコン，10月号，246-255pp.

▶ NEW SOFT
発売予定のゲーム，ヒーロー・オブ・ランスを紹介。――編集部，LOGIN，18号，15p.

▶ SOFT RADAR
最新麻雀ソフト，麻雀狂時代SPECIAL II・冒険篇を紹介している。――編集部，POPCOM，10月号，26p.

▶誌上公開質問状
マスターディスク，デモディスクを手に入れるには？X1turboZでの→，←，↑，↓の入力法について説明，解説している。――編集部，マイコンBASIC Magazine，10月号，66p.

X68000

▶ Mach180をレポート
計測技研が発売した，X68000上でCP/Mを動作可能にするCPUボード「Mach180」についてのレポート。――編集部，マイコン，10月号，347-349pp.

▶ LET's PROGRAMING！
10進数をN（2～9）進数へ変換するプログラムの宿題発表。X68000用，X1turbo用など。――藤本健，マイコン，10月号，246-255pp.

▶ X68000マシン語入門
常駐型プログラムの作成に挑戦。題材はいつでもグラフィック画面をセーブできる「イメージカッター」。――髙橋雄一，マイコン，10月号，350-359pp.

▶なんでもQ&A
X68000用BASICをAUTORUNする方法や，AXに外付けディスクドライブを接続する方法などの質問に答える。――編集部，マイコン，10月号，394-397pp.

▶ MUSIC SQUARE
X68000の音楽環境を，市販されているソフトを中心に紹介する。――編集部，ASCII，10月号，277-282pp.

▶ AV WORKSHOP
スプライトエディタ「Terazzo」とスクリーンエディタ「Xe」を評価。――中山進，ASCII，10月号，336-339pp.

▶ DSH
Human68k上のX形式ファイル用セミオートディスアセンブラ。――川本琢二，ASCII，10月号，354-355pp.

▶ blook
ソフトウェアキーボードに使われていたVRAMを使って，テキストファイルの表示を行うユーティリティ。――中山進，ASCII，10月号，356p.

▶ MIDIライブラリ
外部音源を制御するためのMIDI拡張関数。――市原昌文，I/O，10月号，136-141pp.

▶パターン・エディタ「ぱたぱたくん」
65536色モード対応パターンエディタ。I/Oにおいてすでに掲載された拡張BASICが必要。――WIZARD N，I/O，10月号，217-227pp.

▶ REDUMP.X
見たいときにレジスタの値が見られる常駐型プログラム。――X68K＆浅香唯♡の苦死魔，I/O，10月号，228-232pp.

▶ NEW SOFT
新着ソフト，38万キロの虚空を紹介。――編集部，LOGIN，18号，17p.

▶ X68000新聞
シャープから発売の100インチ液晶ビジョン，グラフィックエディタPRISM-68K，ミッド・ガルツ，ローグ・アライアンス，デジタルクラフト，ねじ式，サイクロンExpressを紹介。――編集部，LOGIN，18号，140-145pp.

▶先取りおすすめゲーム
発売予定の「斬［ZAN］陽炎の時代」を紹介している。――編集部，テクノポリス，10月号，6-9pp.

▶ GAMING WORLD
新着ゲームのファンタジーゾーン，ニュージーランドストーリー，ウイングスを紹介。――編集部，テクノポリス，10月号，17-26pp.

▶新作ゲーム先取り！　SOFT FLASH
フラッピーII・ブルースター復活の日，遙かなるオーガスタ，ジャック・ニクラウス・チャンピオンシップ・ゴルフ，琉球を紹介。――編集部，テクノポリス，10月号，27-29pp.

▶ゲームがオレを呼んでいる！
ロボットアクションゲーム「ジェノサイド」の攻略法と，アドベンチャーアクションゲーム「ニュージーランドストーリー」の攻略テクニック，アイテムを紹介。――編集部，POPCOM，10月号，64-67pp.

▶ X68000ワールド
新着ゲームのウイングス，ファンタジーゾーン，ジャック・ニクラウス・チャンピオンシップ・ゴルフ，リングマスターや，開発中のスーパーハングオン，サンダーブレード，そしてDōGAのCGAシステムを紹介している。――編集部，POPCOM，10月号，83-87pp.

▶誌上公開質問状
X68000用数値演算プロセッサボードの解説や，フロッピーディスクから立ち上がるようにするためのSWITCHコマンドの設定法，EXPERT/PRO用のハードディスクをACEシリーズに内蔵できるか？　について答えている。――編集部，マイコンBASIC Magazine，10月号，66p.

▶ゴルフ・ゴルフ・ゴルフ
トラックボールを使ったゴルフゲーム。――株式会社マイケル商事，マイコンBASIC Magazine，10月号，172-174pp.

▶迷子のピーチャン
森林浴に来て迷子になってしまったピーチャンを，一定距離を進めてクリアするスクロールゲーム。――荻野和弘，マイコンBASIC Magazine，10月号，175-177pp.

▶バイオミラクルぼくってウパ
コナミのファミコンゲームミュージックプログラム。――川野俊充，マイコンBASIC Magazine，10月号，187-190pp.

▶チャレンジ！　X68000
新着ゲーム，ミッド・ガルツ・ゴールド68K，リングマスターを紹介している。――佐久間亮介，マイコンBASIC Magazine，10月号，276-277pp.

ポケコン

PC-1600K

▶ポケットコンピュータ活用術
ポケコンの電子手帳化第5回。今回は時計機能プログラムが登場。――塚田洋一，マイコン，10月号，334-338pp.

PC-1450

▶ EXCITINGプロレス1450
9種の技で10人の対戦相手を倒せ！　PC-1261用プロレスゲームの移植版。――L-L №27NKY，I/O，10月号，208-209pp.

PC-E500

▶誌上公開質問状
PC-E500でのスクロール，POINT命令を用いたドットのチェック法などについて解説。――編集部，マイコンBASIC Magazine，10月号，65p.

▶ PICK UP STONES
全30面の思考型ゲーム。移植版。――MARBOW，マイコンBASIC Magazine，10月号，180p.

オブジェクト指向への招待

オブジェクト指向とは，プログラミングにおける新しい考え方で，Smalltalk-80言語から広まってきた。この考え方は，プログラミングや設計のための方法論，人工知能（AI）分野におけるさまざまな技術などで最近注目されている。本書はオブジェクト指向を「思考を表現するための技法」として捕らえ，基本概念から応用までをプログラミング未経験者にもわかるように紹介している。

富士ゼロックス情報システム㈱/春木良且著
啓学出版　☎03(233)3731　A5判　192ページ　2,000円

スーパーコンピュータ時代

今やあらゆる分野で利用されているスーパーコンピュータ。本書は，その活用に合わせた基礎的な情報を明らかにするために書かれたものだ。スーパーコンピュータの利用者でなくてもわかりやすいように，読み物としてまとめられている。人間にできないことをさせるスーパーコンピュータを使ってのさまざまな開発・利用状況や，さらに使用している人々をもレポートしている。

シドニー・カーリン/ノリス・パーカー・スミス著　那野比古訳　HBJ出版局　☎03(234)3911　B5判　332ページ　2,400円

書籍の価格は消費税込みです

QUESTION and ANSWER

パソコンで使えるファイルの種類にシーケンシャルファイルとランダムファイルがあることをマニュアルで読みました。いったい前者と後者はどのような特徴と欠点をもっているのでしょうか。あつかましい質問ですができるだけ詳しく教えてください。

大阪府　吉田　進

シーケンシャルファイルは順編成ファイルとも呼ばれるもので，基本的にデータをファイルの先頭レコードから順番にしか読み書きすることができません。ランダムファイルは直接編成ファイルとも呼ばれ，データを書き込んだ場所を記録した領域を別に用意しているので，特定の位置に書き込んであるデータをも直接に読み書きすることができます。

ランダムファイルでは，そのファイル編成の性格上，結果的にシーケンシャルファイルと同じ記録方式にすることも可能です。記録媒体としてテープを使った場合には直接編成ファイルを扱うことは難しく，基本的に順編成ファイルしか作成することができません。

順編成ファイルはファイルの編成方法がもっとも単純で，レコードが物理的に連続しているので，記憶領域の利用効率が大変よくなっています。だからテープは当然として，ディスクの空きエリアも無駄なく使用することができます。プログラムやドキュメント，実験データのように，順番に処理できるものは，この形式で扱うのに向いているといえます。また，連続したレコードに書き込むのでデータの読み込み，書き込み時のヘッドの移動も少なく，大量のデータの高速な入出力処理ができます。

欠点としては，特定のレコードをいきなり読み書きすることができないので，データを追加削除などして変更した場合には，ファイルのすべてを読み込んで（もちろん修正する必要のない部分も），更新したファイルを新たに更新ファイルとして作成する必要があります。

これに対してランダムファイルではデータの読み書きが1レコード単位（多くの場合256バイト）ごとに行われるので，追加削除だけではなく，特定のデータの修正変更も簡単に行うことが可能です。すべてのデータを読み出してそれをもう1回書き込むという作業は必要ありません。そういう点でランダムファイルはシーケンシャルファイルより扱いやすくなっています。

シーケンシャルファイルでは一般にデータを保存するフロッピーディスク，メモリなどに加えて，キーボードやディスプレイもファイルとして扱うことができる場合もあります。こう聞いて「キーボードやディスプレイがファイルだなんておかしい，これらはI/O装置だ。ファイルってのは，もっと抽象的なものなんじゃないか」と思われる方もいるかもしれません。しかし，ここでもう一度「ファイル」という言葉をとらえなおしてみましょう。

もしファイルを60字以内で説明せよ，という問題が出されたら，あなたは自信をもってこの質問に答えることができますか？

ファイルとは目的とするデータ処理に適した方式で，同種のレコードを補助記憶媒体や入出力媒体の上に組織的に配列したもの，と定義されています。ですからキーボードもディスプレイもファイルといっても，決して間違いではないのです。

話が横道にそれてしまいましたが，一般的に住所録などデータの追加・削除が頻繁に行われるのであればランダムファイルが，またそうでなければシーケンシャルファイルを使うほうがいいでしょう。

と，ここまでが一般的なパソコンでファイルを扱うための話。X68000を使っている方はおわかりのように，X-BASICのマニュアルなどではランダムファイルとかシーケンシャルファイルとかいった言葉は，まず出てきません。

X-BASICでは従来のランダムファイルで必要だったレコード番号というものがなくなり，1バイト単位でどこでも自由にアクセスできます。1バイト単位にアクセス可能ですから，シーケンシャルファイルとまったく同じような扱いも可能ですし，1行単位や配列を丸ごと読み書きするようなことも簡単にできます。

ランダムファイルなどはレコード単位のアクセスという制限があるため独特なテクニックが要求される分野でしたが，X68000のようなファイル管理ではメモリ上の配列とほとんど変わらない自由なプログラミングができます。X-BASICでこういった形式のファイルを扱う場合の注意などについてはC言語関係の書籍を参考にするとよいでしょう。

編集室のみなさん，こんにちはX68000をたくさんの人が利用していることは同じシャープユーザーとして喜ぶべきことですが，最近はX68000の記事ばかりが表に出ているので寂しい限りです。ところで，私は最近BASICの勉強も兼ねて，簡単なゲームを作っているのですが，そのなかで文字をPCGで表示したいと思ったのですが，AからZまで26個ものキャラクタのフォントをいちいち作成してDEFCHR文で列挙していくのは大変な作業なのです。この手間をなんとか省くよい方法はないでしょうか。なにぶん初心者なのでよろしくお願いします。使用機種はX1turboZⅡです。　**埼玉県　土岐　佳代**

最近本当にOh!Xにも女性読者が増えてきましたね。男性の私としては嬉しい限りです。さてご質問にあるとおりX1シリーズにはプログラムを組む人が自由にキャラクタを定義することができるPCG(Programable Character Generater)RAMが標準装備されていることは皆さん周知のことでしょう。

いまさらあらためて書くほどのことでもありませんが，PCGは基本的に8×8ドットで構成されていて(X1turboでは8×16ドット，16×16ドットもある）ドット単位に8色を指定することができるものです。テキスト画面上に表示されるため処理が速く，高速な処理を必要とする画面スクロールの背景や，大きなキャラクタなどをPCGで書いておくと，プログラムを作るうえで大変楽です。あの名作といわれるゼビウスがX1に移植できたのはPCGがあったおかげともいわれたほどです。

さて，実はこの質問の答えは非常に簡単です。というのもX1turboには文字フォントデータを読み込む命令がちゃんと用意されているからです。マニュアルをひととおり見渡していればすぐ気づくはずなんだけどな……。まっ，初心者ということと女性ということで許してあげちゃいましょう(なんのこっちゃ)。というわけで，土岐さんもすでにこの命令を見つけているかもしれませんね。ただし，ROMに入っているふつうの文字を使うのであればPCGを使うまでもありませんから，ゲームによく使われている太めの文字フォントと，斜体の文字フォントを作る方法をプログラムで紹介しましょう。

リスト1

```
100 '
110 ' PCG ﾃｲｷﾞﾌﾟﾛｸﾞﾗﾑ
120 '
130 PRINT "1.ﾌﾂｳ"
140 PRINT "2.ﾌﾄｼﾞ"
150 PRINT "3.ｼｬﾀｲ"
160 PRINT "ﾄﾞﾚﾆｼﾏｽｶ"
170 REPEAT
180 A$=INKEY$(1)
190 UNTIL A$>="1" AND A$<="3"
200 ON VAL(A$) GOSUB "FUTSUU","FUTOJI","SHATAI"
210 END
220 '
230 ' ﾌﾂｳ
240 '
250 LABEL"FUTSUU"
260 FOR I=&H20 TO &H7A
270   A$=LEFT$(CGPAT$(I),8)
280   DEFCHR$(I)=STRING$(3,A$)
290 NEXT
300 RETURN
310 '
320 ' ﾌﾄｼﾞ
330 '
340 LABEL"FUTOJI"
350 FOR I=&H20 TO &H7A
360   A$=""
370   FOR J=1 TO 8
380     A=ASC(MID$(CGPAT$(I),J,1))
390     A=A OR (A*2 MOD 256)
400     A$=A$+CHR$(A)
410   NEXT
420   DEFCHR$(I)=STRING$(3,A$)
430 NEXT
440 RETURN
450 '
460 ' ｼｬﾀｲ
470 '
480 LABEL "SHATAI"
490 FOR I=&H20 TO &H7A
500   A$=""
510   FOR J=1 TO 8
520     A=ASC(MID$(CGPAT$(I),J,1))
530     IF J<3 THEN A=INT(A/2) ELSE IF J>5 THEN A=(A*2 MOD 256)
540     A$=A$+CHR$(A)
550   NEXT
560   DEFCHR$(I)=STRING$(3,A$)
570 NEXT
580 RETURN
```

リスト1がそのプログラムです。CGPAT$，DEFCHR$の機能がよくわからない方はマニュアルを参照しながら読んでください。

ではプログラムを簡単に説明しましょう。250行から300行まではCGROM（文字フォントが書き込まれているROM）から8×8ドット構成のキャラクタコードを読み込みます。1文字は8バイトのフォントデータで構成されます。そしてそれを，青，赤，緑のすべてのページに書き込むように，280行で8バイトのデータを3回繰り返して24バイトのデータに加工します。で，それをそのままPCGに定義しています。これだとCGROMの内容をそのままPCGに定義したことになります。

340行から440行までがふつうの文字より太めのフォントを作る部分です。380行で文字フォントデータをひとつずつ（上から1ライン分）取り出します。390行でいま取り出した文字フォントを右に1ビットシフトして，シフト前のデータと論理和をとります。ここがミソです。

たとえば，

○○●○○●○○（●が表示される）

というフォントを例に取ってみましょう。このフォントで表示される点を1，そうでない点を0として2進数で表すと，00100100となります。これを左に1ビットシフトすると01001000となり，この2つのデータの論理和は01101100で実際のイメージだと，

○●●○●●○○

となります。この処理を8ライン分繰り返して1文字のフォントデータができあがります。実際にいろいろなフォントデータで紙の上で実験してもらえば，その様子がよくわかるでしょう。

さらに先に進みましょう。480行から580行が斜体文字のフォントを作る部分です。太字のフォントを作るプログラムと違う部分は530行1カ所だけで，ほかはすべて同じです。530行で行っている処理は文字フォントデータが上から3ライン目より上だったらデータを右にシフト，5ライン目より下だったらデータを左にシフトさせています。これも実際に紙に書いてやってみるとよくわかると思います。

以上，多少わかりづらい説明だったかもしれませんが，PCGで文字を定義するうえでの参考にしてもらえればうれしいです。なにかプログラムができたらぜひとも送ってくださいね。

えー，突然ですが今月はスペースが残っているので（要するに取り上げたくなる質問が少なかったのであるが）ちょっとひと言。毎月毎月たくさんの質問をお寄せいただきありがたいのですが，いまでも質問の内容がよくわからなかったり，（今月は取り上げましたが）マニュアルを見ればわかってしまう質問などが結構あります。また，「最近ディスプレイにノイズが出るのですが修理に出したほうがいいでしょうか」とかいった，ハードウェアの異常に関する質問は質問箱に相談するよりもシャープのサービスセンターに問い合わせたほうが確実です。

それと選ぶ私の意見としては，葉書にちょこちょこと質問を書いて威張っている文章より，詳しい内容を封書で送ってもらったほうがうれしいのです。このページの右下にも書いてあるけど，質問内容が図などを使って説明できる性質のものならば必ず同封してください。どうしても取り上げてほしい質問があったら「絶対載せてください！」って赤ペンで書かなくていいし，字だって汚くても読めればいいから，聞きたいことを「もう頼むからそこまでいわないで」とこちらが思うくらいに詳細に書いていただければ私としてもそれなりに対処しますので，どんどん質問をお寄せください。よろしくお願いします。

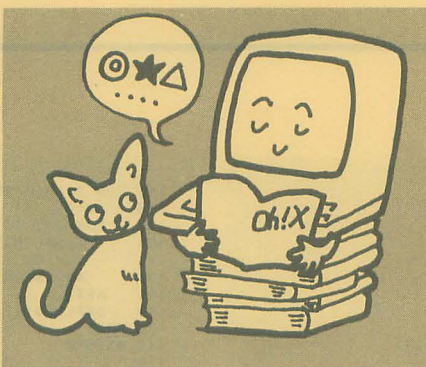
最近の傾向ではX1関係ばかり取り扱ってますが，もちろんX68000やMZ関係についての質問も随時受け付けています。というわけで，来月までごきげんよう。

（影山　裕昭）

質問にお答えします

日ごろ疑問に思っていること，どんなことでも結構です。どんどんお便りください。難問，奇問，編集室が総力を上げてお答えいたします。ただし，お寄せいただいているものの中には，マニュアルを読めばすぐに回答が得られるようなものも多々あります。最低限，マニュアルは熟読しておきましょう。質問はなるべく具体的に機種名，システム構成，必要なら図も入れてこと細かに書いてください。また，返信用切手同封の質問をよく受けますが，原則として，質問には本誌上でお答えすることになっていますのでご了承ください。なお，質問の内容について，直接問い合わせることもありますので，電話番号も明記してくださいね。

宛先：〒102 東京都千代田区
九段南2-3-26井関ビル
㈱日本ソフトバンク出版部
「Oh！X質問箱」係

FROM READERS TO THE EDITOR

読書の秋，スポーツの秋，食欲の秋，パソコンの秋，ついでに人事異動の秋？ 編集部では新しいメンバーを加えて張り切っています。彼らもまた、読者の皆さんの声を頼りにドラゴンへの道を歩むことでしょう。よろしくご指導くださいね。

◆HDの特集，ありがとうございます！　この次は，ビデオ，イメージスキャナ，ポケコン，ジョイカード類まで含んだパソコン周辺機器総ざらえでカタログっぽい特集をぜひ。Oh!Xらしくないイージーな企画かもしれませんが，これからの予想なども行えばよいのではないのでしょうか。周辺あってこそ楽しいパソコンですから。

石井　健　(21)　広島県

清く正しいカタログというのはイージーどころか超大変な特集になるでしょうね。だからやれない？　うーん。

◆ちょうどHDを増設しようと思っていたところなので，グッドタイミングでした。山形のような地方ユーザーには，ソフト・ハードの評価記事は，貴重な情報源です。これからも内容のあるものをお願いします。２カ月前ほどから，中古モデムで，パソコン通信を楽しんでいます。パソコン通信も下火だそうですが，現状など特集してください。　菅井　隆夫　(27)　山形県

X68000関係では面白いPDSとかがネットワークで流れているようですから一度取り上げてみたいですね。結構危ないものもあったりして扱いに困るかもしれませんが。

◆だーもう，HDの特集なんかするから，欲しくなったじゃないか！　どうしてくれる？

寺村　公一　(22)　滋賀県

特集のおかげで編集部の私たちもハードディスクが欲しくなっちゃって……，休みが取れたら秋葉原にでも買いに行こうかなぁ。でも先立つものがねえ。

◆やっとZ'sSTAFF PRO-68K ver2.0を買いましたけど機能がありすぎてなかなか覚えられません。絵を描くなんてとてもとても。今度Z's STAFF PRO-68Kでお絵描き講座やってくれませんか。30のおじさんにもわかるように。

佐藤　廣茂　(30)　宮城県

最初は気に入った絵や写真を写すような気で描くといいと思いますよ。ごく一部の機能だけでも結構使えますからね。

◆会社では端末代わりのFMR60を使い，家ではX68000を使用している。会社では今テトリスを昼休みにやっている人が大勢いる。話は変わりまして，今月特集のハードディスク＆プリンタの活用は，たいへんよくわかりましたのでこれからもこういった特集を載せてください。

三田　恭一郎　(23)　東京都

◆仕事でTOWNSを使っていますが，X68000のほうがプログラムしやすい。PDSも多いし解像度もいいと思い，会社には悪いのですが，X68000を使わせていただいております。

小林　文明　(21)　長野県

会社というのは……，いやいや，そういう方はわりといらっしゃいますよ。でも，TOWNSとX68000を両方使っているなんて，ちょっと羨ましいな。

◆X68000はどれぐらい出ているのでしょうか。ある人の話では，７万台近くまで出ているとのことですが，実際どのくらいか教えてください。また，ソフト（特にゲーム）はどれくらい売れてるのでしょうか。それに，最近ゲームに飽きてきたが，ほかの人はどうか知りたい。でも，プレゼント商品にジェノサイドを希望しているのが，ミソなのだ。石川　立城　(23)　埼玉県

X68000の出荷台数は，今年の８月現在で７万6500台にのぼっているそうです。来春にはいよいよ10万台を超しそうですよ。

◆THE COMPUTER９月号の「編集部から」で“シャープがもう一つのパソコン文化の担い手を標榜した”と書かれていた。X68000のことだが本当にそう思う。X68000を買った方はシャープと共に挑戦の道を歩んでほしいものだ。それにしてもMZ....MZ，ZZZZ...

大森　基弘　(18)　滋賀県

THE COMPUTERでは，挑戦的存在ともいえるパソコンX68000に注目しているようですね。

◆とうとう「スターシップ・ランデブー」がX68000に出てしまった。僕はPCユーザーのところへ通って，あの裏技を使って，最後までやったんですけど，X68000版もあの技は使えるのでしょうか？　もし使えるとすれば，それだけで買う価値はあるかも。

小宮　崇　(18)　埼玉県

◆TETRISもいいけど，じっくり考えるPITMANもなかなか。最近はRPGをやる気力もなくなり，シューティングをやる体力もなくなった。やはり，行き着く先はパズルなのか。

平木　敬太郎　(21)　福井県

◆サスケ，スイカソーダ２……。いったい誰が仕入れてくるのだろうか？　それにしても近寄り難い雰囲気のバーだなぁZ80's Barって。
P.S.誰か「リゲイン」のコマーシャルソングをLIVEに投稿しないかな。

尾形　秀晃　(18)　千葉県

ウン，いいところに気がついたね。でもリゲインの黄色と黒ってソフトバンクのロゴカラーといっしょでなんか恥ずかしいなぁ。

◆P1.EXEやZ'sWord JGのような高機能ワープロが早く出ないかなぁ〜(某太郎はダメ！)。ところで，OS-9の初心者向け連載をお願いします。　郡　茂樹　(33)　兵庫県

◆最近思ったことですが，最近のパソコン誌は，やたらゲームやその他ソフト関係ばっかりで，自分でプログラムを組むという傾向がうすいようです。Oh!Xはその点で安心ですが，その傾向を作り出せる，メッセージをもった，20歳を過ぎても読める雑誌であってほしいと思います。

◀上田　修(19)岐阜県
8月号に載ったX-700プロジェクトのイラストですね。それにしても，AV指向の高水準S-BASICというのがMZを知る者には笑える。

◀小松　恭郎　長野県
待望のX68000ラップトップじゃぁ！と思ったけどキーボードのサイズから察するとこれってかなり大きくありません？

南波　昌幸　(23)　千葉県

プログラムの楽しさを伝える雑誌がなくなったらプログラマになる人がいなくなってしまう。そうしたらいったい誰がゲームを作ってくれるのだろうか？

◆先日，DōGA・CGAシステムが届きました。初めは何をしたらよいのかわからなかったのですが，2，3日システムをいじくりまわしているうちに，ようやく概要がつかめてきました。自分で一からアニメーションを作ろうとがんばっています。DōGA・CGAアニメーション講座が長く続きますように祈っております。

葦原　文夫　(34)　秋田県

◆私のX68000ACE-HDが先日，雷撃を喰らってあえなく即死してしまいました。あの基板が焼けるキナ臭いニオイを私は一生忘れることはないでしょう。あ〜，これから修理に出さにゃ。でも，X68000をつぶしたカミナリが直撃した，近所の家なんか電化製品の全滅はもとより，家の壁がぶっ壊れてたとゆ〜んだから，自然とは恐ろしいものだと思ってしまう今日この頃です。

中山　秀隆　(19)　三重県

これは怖い！　なんとか無事に修理できますように。

◆ファミコンやPCエンジンなどのソフトは人気のある品物でも日数がたてばかなり安くなりますよね。それなのにパソコンソフトときたら信じられないようなことがあります。たとえば○△☆というソフトが定価をそのまま，そのうえ消費税まで取ろーたぁふてぇ奴だ！

岡田　真二　(29)　福岡県

◆日本橋のある店でMZ-2500用のゲームソフト数十本が，それぞれ300〜800円で売られていた。X1の未来を暗示しているようで寂しかった。それから，僕の名前は「陣内」ではなく「陣山」です（9月号155ページ）。

陣山　達夫　(19)　大阪府

古くなったソフトが定価で売られるのは気に入らないかもしれないけど，売れなくて叩き売られるのもねえ。

◆絵を描くのが好きなので，弟のX68000でZ's STAFF PRO-68Kを使ってみました。画像の美しさに驚かされました。パソコンておもしろいですね。　廣澤　かをり　(19)　東京都

◆やっとC Compilerを買った。これからどんどんプログラムを作るぞ！　まずはアクションRPGを作って金をかせいでみせるぞ！

山根　慎二　(19)　岡山県

来月はCの特集です。お役に立てればよいのですが。

◆先日，私の友人Nが廃棄処分になった学校のMZ-80K2Cと山のようなテープを引き取った。MZ-80K2Cとは，そもそも私の学校にはMZ-80CとMZ-80K2のふたつがあって，ジュースづけなどの乱暴な扱いをされたあげく，Cのボディはケトケトに，K2のボードはいかれてしまった。そこである人が考えた！　CのボードをK2のボディに移植したのであった。手術は成功し，友人Nはそのボディにカッコよく“C”のロゴを入れた。こうしてMZ-80K2Cは誕生した（つづく）。　吉池　信悟　(18)　東京都

いるんですねえ，こういうすごいことをやる人が。ぜひとも続きをお願いしますよ。

◆科学や技術に関わる人で夢を持たない人なんているんですかね。夢もなく，ただ必要のみにかられて研究する……。うーん想像もできない。私個人は，人間は条件反射のかたまりだと考えています。すべての欲求の源は生存欲求であり，それを土台として（それの制御を受けつつ），自立的なネットワークを形成しているのではないでしょうか。ちなみに，小学生のころは，状況に対応しようとする関数として考えていました。もっともニューラルネットもそのようなものですが（でもニューラルネットのほうが実現しやすい）。　江原　忠士　(19)　岡山県

ちょおおっと，小学生のときからそんなこと考えていたんですか？　コワ〜い。

◆X68000を購入し早2カ月。ゲーム主体で使ってきたのでプログラミングはまだまだ。そこで，なんでもよいから情報を集めようと思い，初めてOh!Xを買いました。X68000の操り方を適度に載せてくださいませませ。

岩井　清彦　(22)　愛知県

◆パソコン通信やってると，X68000の面白そうなPDSがいっぱい……。早く大学に合格して買いたい！　山下　貴幸　(19)　北海道

◆THE SOFTOUCHを見てファンタジーゾーンに感動して，つい買ってしまった。買って感動，やって感動。これからもよいソフトを取り上げていってください！

石神　覚司　(16)　静岡県

◆HDDレポート。私のX68000にはウインテクのHD202（20MB）がつながっている。OS-9での使用も可。ブレイクキー，コントロールキー＋ファンクションキーでのシッピングも可。

荻原　毅　(24)　東京都

◆初めて買いました。HDとサイバースティックの記事があったので買いました。これからもいい記事があれば買います。

慶野　利幸　(16)　栃木県

HDとサイバースティック！　うーん，パソコンはこうでなくっちゃね。

◀杉本　秀昭　宮城県
これはゲームのイメージイラストかな？　しかし，セーラー服の女の子を泣かせるからには，よほど感動的な物語なんでしょうねえ。きっと。

◀杉浦　豊　大阪府
ニュージーランドストーリーのイラストって多いんですよね。みんなキウイのティキがうまく描かれていて，どれを載せようか迷っちゃった。

◆プレゼントのCDが届いたのでさっそく聞いてみました。とてもPSGでは出ないような音が出ているので僕もFM音源などがほしくなりました（それよりディスクがほしいーっ！）。もっとも，僕はまだPSGの限界に達する音を作ったことがないのでFM音源なんて使ってもどうせたいしたことはできないだろうけど。

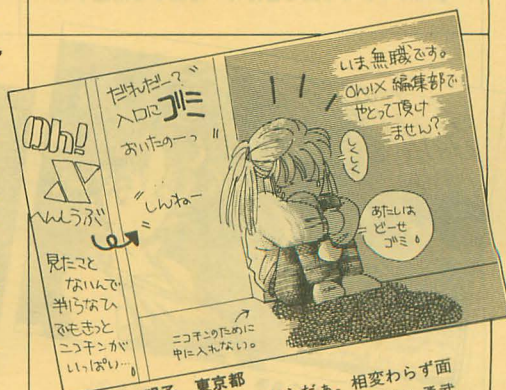
田中　真実　(16)　滋賀県

ゲームミュージックでは今月のメタルホークもかなりのものですよ。

◆簡単な欧文ワープロでもつくってみようかと思っていたところへ先月からの祝さんの記事。タイムリーではあったのですが，やはり自作の参考にするには情報量が少ないようです。ぜひ，データ構造について，特集なり連載なりで取り上げてください。また，参考になるような文献があれば教えてください。

山田　祐一　(21)　東京都

◆「立川くんのよいこのためのFM音源講座」ためになりました。ぜひ第2回もやってほしいです。　稲垣　俊彦　(16)　大阪府

◆「Defeat X」には驚きました。あんなちょっとのプログラムであそこまでできるんですねぇ。いやー，まいった。ぼくなんかマシン語の1つひとつの命令を扱えないんです。うらやましいですねー。早くプログラムが組めるようになりたい。　西村　武雄　(18)　京都府

▲鶴見　明子　東京都
やった一、また鶴見さんからだぁ。相変わらず面白い。ニコチンはそこそこですが、一度偏し勇武に来ませんか。特別に取材（？）を許可しますよ。

◆Defeat Xなかなか面白いですね。最初に走らせたとき奥行きもあるのかと思った。それにしても3面のボスキャラは強すぎる。
白井　透公（18）山梨県

◆Defeat Xはすごいですねえ。昔だったら十分売りものになると思います。でも，3，4面は敵がすごくて，2，4面のボスキャラはむずかしい。あれだけのリストでこんなにいいものができていいのだろうか。敵機の飛行パターンもいろいろあってよいと思う。ただ，バリヤーなどのパワーUPがあればなぁ。
花井　康浩（19）愛知県

久々に本格的なアクションゲームで人気も上々。X1/turboユーザーにもまだまだ頑張っている人がいてくれて心強い限りです。

◆MIDIボードにシャープがMIDI DRIVERをつけなかったのは非常によくないと思う。市販のソフトが，MT-32にしか対応していないなら自分でつくるとなると全部1からやれなんて横暴やね。ところでシャープ純正4chDSPシンセサイザなんて出さへんかな。本体からもデータ送れて。音楽以外のデータも処理できてとかいうやつ。MOディスクもつなげて！
中島　祐治（20）大阪府

◆最近不愉快なこと。もっと先に取っておきたかった“Dyna Book”という名称をその資格のない機械が使ったこと。うれしいことは，X68000に“C”以外の言語“FORTH”が載ったことと，その価格が安いこと。品質と値が比例していなければと心配です。日本語は使えるのでしょうか？　X68000が，次の段階に発展しつつあるのを感じる。さて，次はワープロソフトが欲しい。
三浦　直樹（36）青森県

Dyna Bookというのはとっておきの名前ですからね。

◆趣味であるデータベースを構築していますが，過去にさかのぼってそのデータを入力するのに長大な時間がかかります。X68000と完全互換で超小型超軽量4電源方式のラップトップコンピュータは出ないのでしょうか。外出先で，仕事先での少しの時間をも活用できればすばらしいのに。
林　謙治（30）大分県

◆私は今まで1回も自分の「推薦する市販ソフト」が「読者が選ぶ今月のゲーム10」に載ったことがない。だって最近のは，やってないのばっかだもん。にもかかわらず，今月も私はファンタジアン（アドヴァンストではない）に1票を入れる。
小口　卓（16）千葉県

なるほど，好みの問題ではないようですね。

◆パソコン雑誌は高3になったら皆やめたる！と心に誓ったのはもう半年前。夏までにOh!Xもやめるぞと思ったのは2カ月前。夏休みがすべてだ，Oh!X読むのは今度こそやめだ！　と思ったのは3日前。嗚呼，また買ってしまった。こりゃ麻薬だ。理系を選択してしまって，びぶんせきぶんに苦しむ生活をおくる毎日になってしまったのも，元はといえば皆Oh!Xに出会ったからだ。責任とって僕にプレゼントください。
南口　経昭（17）大阪府

◆以前ソーサリアンの広告で「自分でシナリオをつくってしまう」ツールを出すようなことを書いていた。どうなったのだろうか。ファルコムさんだからシナリオ自体の流れはTINY-BASICのような形なのかもしれないし，グラフィックやサウンドのおいしいエディタをつけてくれるだろうし，それでお値段3,800円とかしてくれるんだろうなぁ。木屋さんは今何を作っているんでしょう。まさか温泉で遊び呆けてるわけでもないだろうし。僕はこれが望みでソーサリアンを買っているのだぞ！　その裏にはグラフィックやサウンドツールに恵まれないX1ユーザーの悲しさがある。ワープロつけてくれてもありがたいなぁ。ってなんだよこのあつかましさは。
馬場　啓示（16）宮城県

友達同士で相手の作ったシナリオで遊べるようになると面白そうですね。ソーサリアンのようなシステムならそういった可能性もあるように思うのですが。新しい，ゲームシステムとして実現しないかなぁ。

◆村田氏のマシン語講座ですが，えらく気合いが入ってますねェ。X68000のユーザーのレベルを上げようという意気込みが感じられます。これからもバカな（？）意見に反してがんばってください。
松浦　信生（17）神奈川県

◆とうとうFORTHが出た。どっちを向いてもCばかりの世の中に喝を入れてくれたようだ。こんどは1日もはやくCOBOLとFORTRANが欲しい！　そりゃぁ，Cに比べればマイナーかもしれませんけれど，流行になってなくてもいいものって，あるんですから。オレは流行なんてでぇっきれぇなんだ！
国政　寛（18）大阪府

FORTHが好きな人ってユニークな人が多いようですね。

◆ぼくはハードディスクをおまけみたいなのでもっているんですけど，西川善司さんや荻窪圭さんの記事を見てぼくのACE-HDは半分も活用されていないように思いました。こんどの特集は本当に役に立ったと思っています。それでいろいろやってIPLがどれを立ち上げるかを聞いてくるというのが知らなかったのでやってみたんですけどよくなりません。西川さん，あれはHDの中のOSのバージョンは同じでないといけないんですか？　それともACE-HDだからならないの？
牛島　睦（17）福岡県

◆ジェノサイドは難しかった。未だに3面の後半につまずいている。ファンタジーゾーンは80面まで進んだけどもう変化はないのだろうか。試験が近いのにこんなことをしていてよいのだろうかと思う今日このごろです。
堀内　敏（20）北海道

◆今月のプレゼントにX1/turbo用のソフトがなかった。くやしいから，ジェノサイドにして倍率を上げてやる。
本郷　慎一（16）京都府

◆今まではNECのPC-88を使っていたけど，今年6月にX68000に買い換えたところです。が，Oh!PCに比べると値段が高いのが，買うたびに気になります。厚みは広告の量が違うのでわかりますけど，定価がOh!PCより高いのはゆるせません。広告料をとっているなら，それをキープして，Oh!XやOh!FMにも振り分けて，せめて定価を同じにしてください。まだ3回しか買っていないのに，態度がでかいのはわかっています。
斎藤　義知（19）大阪府

いや～，建設的なご意見ですね。ぜひともうちの社長や局長にも見せてあげたい。でも，Oh!XがPCの稼ぎの恩恵を受けるなんてシャクじゃありませんか？　今のOh!Xは独立した雑誌として存続できるわけですから，これからも応援してください。

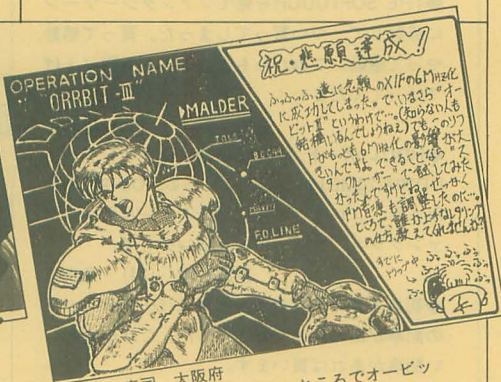
◆Oh!PCが月2回発刊されるTVコマーシャルを見たが，Oh!Xのサブリミナルメッセージでも入ってたのではないでしょうか？　早く18日がこないかなぁと思う今日この頃。
沼部　栄士（20）群馬県

実はひとコマおきにOh!Xの表紙が埋め込まれて……，んなわけないって。

◆X68000のVRAMパレットについての記事をわかりやすく載せてください。X68000専門の本（『X68000テクニカルデータブック』アスキー，Oh!MZ 86-12月号）を見たけれどさっぱりわからない。
日沖　則幸（18）東京都

X68000のパレットはちょっとわかりにくいですよね。

◆1学期に偏差値を10以上も落とし，夏休みも

▲田村　憲生（20）広島県
田村君お久しぶり，最近はイラスト常連組も少しずつ入れ替わってきたでしょう。でも，Oh!Xでは古参の一人として頑張ってくださいね。

▲宮本　康司　大阪府
6MHzですか，やりましたねえ。ところでオービットIIIはいいゲームだったとU氏も高く評価してるんですよ。6MHzならきっと速いだろうなぁ。

遊びまくった危機感のまったく感じられない菅原君からのレポート。題して，
——大学合格のために——
1) 親に合格したらX68000を買ってもらう約束をする。
2) X68000のカタログを勉強部屋に貼る。
3) くじけそうになったらカタログを見て「絶対に手に入れちゃる」と机に向かう。
4) 大学に合格する。
5) X68000が我が手中に。
もちろんこの場合，バイクでも車でも家でも兵器でもかまわないことは言うまでもない。
菅原 裕輔 (17) 岩手県
いやはや壮大な計画ですね。X68000を心の支えに頑張ってくださいね。

◆X68000のHD環境でまだ足りないものがあります，それは98のエコロジー，ノストラダムス，ファイルマスターなどのようなものです。特にX1からステップアップするようなときにはファイルマスターのCP/M⟷MS-DOSコンバータは必要不可欠です。あとMIFESがあれば買うのになぁ。 仲田 宏生 (22) 岡山県
HDの中身を整理するのってけっこう大変ですからね。

ぼくらの掲示板

●掲載ご希望の方は，官製ハガキに項目(売る・買う・氏名・年齢・連絡方法……)を明記してお申し込みください。
●ソフトの売買，交換については，いっさい掲載できません。
●取り引きについては当編集室では責任を負いかねます。
●応募者多数の場合，掲載できない場合もあります。

仲間

★パソコンよろずクラブ「すいか・くらぶ」では，ただいま機種を問わず会員を募集しています。情報交換と会員間のコミュニケーションを目的として活動。初心者から上級者まで楽しめる月1回発行の会誌にはMS-DOS，D&D，RPG，アニメ，ビデオ，イラストなどの多種多様な話題でいっぱいです。パソコン通信を利用した活動もしています。入会案内書を72円切手同封でご請求ください。翌日には発送いたします。 〒328 栃木県栃木市今泉町2-8-67 飯塚昌弘「NG」係

★「あなたのパソコンライフを100倍楽しくする」をキャッチフレーズに「FRIEND'S」では会員を大募集します。機種はXシリーズをはじめ何でもOKです。2カ月に1回ゲーム中心の会報を発行します。会員を失望させない息の長いサークルを目指します。詳しくは62円切手同封の上，連絡を！ 〒038-37 青森県北津軽郡板柳町常海 橋稲葉61 川口昭男(28)

★「Computopia 68ｋ」第2期会員募集。対象機種は，X68000です。活動内容の主なものは，月1回発行の会報です。とにかくにぎやかにやっていくので，多くの会員の募集をします。詳しいことが知りたい方は，62円切手2枚と自己PRを同封の上，下記まで連絡を。〒443-01 愛知県蒲郡市西浦町女松山146-2 青山敦男(16)

★X1turboシリーズユーザーを対象としたサークル「秘密結社電脳組」を発足させるにあたり，会員を募集します。活動は，月1回のディスク会報を中心に行っていきたいと思っています。「プログラムはバッチリだぜ！」という人や「イラストならまかせろ」という人，「オレはマニアだぜ」という人，「私，初めてなの」という人，すこしでも興味をもたれた人は，(1)62円切手を貼った封書を同封の上ご連絡を。折り返し入社案内をお送りします。(2)300円分の為替同封の上ご連絡を。折り返し「マボロシの創刊号ディスクマガジン1号」をお送りします。 〒079 北海道旭川市永山5-3 斎藤隆博(20)

★パソコンサークル“HIRORIN”では活動の中にS-OSをサポートすることになりました。内容はリストの共同入力やPDSの配布です。興味のある方は62円切手同封の上，ご連絡ください。 〒190 東京都立川市柴崎町2-17-27 道アパート10号 細野信明(27)

★X68000ユーザーを対象としたサークルの会員を募集します。活動内容は，プログラムの情報交換やPDSの交換です。詳しいことは，62円切手同封の上，封書にて連絡を。 〒214 神奈川県川崎市多摩区菅城下20-6 須川雅志(17)

売ります

★X1用カラーイメージボードCZ-8BV1を1万5千円で。それとFM音源ボードCZ-8BS1を8千円で（どちらも箱付き，マニュアル，ケーブル，付属品付き)。連絡は往復ハガキで。 〒351 埼玉県朝霞市本町1-22-35 遠藤克之(17)

★MZ-1P07にトラクタユニットMZ-6P09をつけて，送料別2万円で。インクリボンもつけます。また，MZ-1P17用第2準ROMを送料別7千円で。連絡は，往復ハガキで。 〒739-17広島県広島市安佐北区落合南4-41-6 小野靖弘(19)

★MZ-1500用RAMボード・漢字ROM（マニュアル・箱なし）を各々5千円以上で。X1用FM音源ボードCZ-8BS1（マニュアル・ソフトあり，箱なし）を1万円以上高い人優先で。すべて手渡し希望です。連絡は，往復ハガキで。 〒273 千葉県船橋市海人4-1-17 赤城弘満(16)

★漢字プリンタCZ-8PK5（完動，箱，インクリボン2コ付き）を6万円で。連絡はTEL明記の上，往復ハガキにて。 〒604 京都府京都市中京区東洞院通御池上ル船屋町423 プレシャス御池403 西川裕幸(18)

★X68000用MIDIボードCZ-6BM1と，ローランド音源モジュールのD-110をセットで5万円で。付属品はすべてあり。両方ともS63.12購入。単品も可。なおセットの人はオマケ付きです。連絡は，TEL明記の上往復ハガキにて。〒532 大阪府大阪市淀川区西三国1-7-2-502 松浦琢磨(18)

★X68000用I/O拡張BOX CZ-6EB1を4万円，イメージユニットCZ-6VT1を3万5千円，2MBRAM CZ-6BE2を5万円。いずれも，箱，保証書，取説付き。連絡は，往復ハガキにて。 〒399-07 長野県塩尻市広丘郷原1762-286 野崎潤(20)

★X1用FM音源ボードCZ-8BS1（箱，マニュアル，付属品付き，完動）を送料込みで1万円で。連絡は往復ハガキで。 〒064 北海道札幌市中央区南4条西7丁目1番地 高橋和宏(17)

★電子システム手帳PA-7000（箱，マニュアル付き）と，電訳機英和・和英カード，PA-7C1をあわせて1万円前後で。連絡は往復ハガキで。 〒498 愛知県海部郡弥富町稲元字起畑5 尾内司(17)

★ディスクドライブCZ-503F（箱，マニュアルその他付き）を1万9千円で。往復ハガキで連絡を。 〒376-01 群馬県勢多郡新里村新川548-6 小沼克彦(16)

買います

★X1用FM音源ボードCZ-8BS1を8千円で。NEW-BASIC CZ-141SFを8千円で。また，MT-32を3万円で買います。連絡は往復ハガキで。 〒238-03 神奈川県横須賀市長井3-1-9 小熊靖則(18)

★X68000用カラーイメージユニットを3万円前後で。マニュアル，付属品つきで傷・汚れは可。大至急お願いします。連絡は官製ハガキで。 〒916 福井県鯖江市吉江町12-9 佐々木正樹(17)

★プリンタCZ-8PC3か4を2万5千円前後で。連絡は往復ハガキで。 〒921 石川県金沢市米泉町3-42-5 吉田誠(16)

★カラーイメージボード2 CZ-8BV2を送料込み2万円ぐらいで。また，ディスプレイテレビCZ-830D，CZ-860D，CZ-880Dを送料込み4，5万円で。CZ-850D，CZ-855Dは3，4万円。連絡は往復ハガキで。 〒982 宮城県仙台市太白区八木山弥生町23-23グリーンハイツ203号 菊池淳(20)

★MZ-2500用カラーパレットボードMZ-1M10を4千円以下で(送料込み)。完動品なら箱，説明書の有無は問いません。連絡は希望価格明記の上，往復ハガキで。気長に待ちます。 〒565 大阪府吹田市山田西3-21 千里レックスマンションA棟110号 中城康伸(19)

バックナンバー

★Oh!MZ 1985年3月号を送料込み千円で。切り抜き不可。連絡は往復ハガキで。 〒948-02 新潟県中魚沼郡川西町赤谷 高橋直人(36)

★1985年12月号，86年1～2月号，6～11月号を各千円（送料込み）で。切り抜き不可。連絡は往復ハガキで。 〒860 熊本県熊本市本山4-8-10 大岩雅裕(25)

編集室から from E・D・I・T・O・R

DRIVE ON

このコーナーでは，本誌年間モニタの方々のご意見を紹介しています。今回は，9月号の記事に関するレポートです。

●ハードディスクの広大な容量はとても魅力があるのですが，フロッピー以上にデリケートなため，振動に極端に弱くクラッシュしやすい点が非常に嫌ですね。またディスクそのものが出し入れができないので，さらに容量が必要なときには大きなものに換えるか，増設するしか手がなく，金銭がかかってしまいます。ミニコン用のもの（φ80cmぐらいの巨大なもの）はディスクをフロッピーのように交換して使用するタイプがありますが，パソコン用もそんなタイプが出現すれば便利になると思います（ただし，取り扱い上の注意が必要ですが………）。もっとも，2，3年しないうちに光磁気ディスクにその立場を譲ることになるでしょう。
藤原博人（25）X1turboZ　鳥取県

●「7機種接続＆総チェック」を読んで，X68000にとってHD環境はまだまだ厳しいものがあると感じました。外付けHDではitecのITXを使っている人が多いのではないかと思う。が，どうも性能的にはイマイチの感じがする。表1は購入予定者にとっては手っ取り早い一覧表として有益だと思う。ただ，定価と実際の価格の間にかなりギャップがあるように感じられたので，参考として秋葉原とか日本橋での販売価格も載せてあればもっと実用的選択の参考になると思った。
森川一（24）X1turboII，X68000ACE-HD
北海道

●「ハードディスク雑学講座」はよかったです。できたらFDDと比較して記述するとなおメカニズムに対する理解を深めることができたでしょう。セクタNo.やトラックの振り方が，一般国内メーカーとIBMなど米国メーカーで違ってたりすることや，HDDの生い立ちをもう少し詳しく書いてほしかった。大型機などで採用している空気噴出式のディスク走査や，カートリッジ式のメカについても，もうちょっと書いてほしかった。
中野賢一（29）MZ-2000，X1/G，X1turboII，FM-8，FP-200，PC-8801/9801，PC-1251，B16LX
山口県

●「空間有効利用の心得」を読むまでハードディスクは「分割して統治」するものなのだとは知らなかった（もっとも私の場合ほとんどが知らなかったとなる）。それに，使い込むと遅くなるという謎も解き明かされて，ああためになったな，という感じがする。実際にハードディスクを使うようになったら，十分参考にできそうである。
山中伸之（31）MZ-2500　栃木県

●以前にプリンタスプーラ機能が発動しているときにCOPYキーを押したあと，すぐにプリンタの電源を切ってしまったとき，実に困った覚えがあるので，この「COPYキーメニュー」はかなり役に立つプログラムではないかと思います。それに「COPYキー」を「CTRL＋ファンクションキー」に変更してメニューをもう少し増やすだけでプリンタバッファクリア機能だけではなく，もっとほかの機能が（たとえばOPMのコントロールやカレンダーに時刻を表示など）すぐに拡張できそうなので一度試してみようと思いました。
田中実（19）X1turboII，X68000ACE　大阪府

●アナログジョイスティックはそれなりに面白そうではある。GAMEなど（特に精度を要求される方面）には，きっと欠かせないものになるやもしれません。記事でもそういった将来性も含めてなかなか詳しく（ある意味ではもの足りないが）書いてあるし，有望性をほめてある。ちょっと話は変わるが，バイクではアクセル（つまりはフューエル調節）に従来のアナログ（ワイヤー）をデジタルに変えたものが出てきた。コンピュータ界ではアナログを取り入れ，モーターサイクル界ではデジタルを取り入れている。うーむ，興味深い。ほかのアナログ装置が出てくるのがとっても楽しみである。
大津和之（19）X1turboZ　福岡県

●「X68000マシン語講座」では，毎回感心することばかりです。今回もこうした連載にありがちな「ここではこうなっている」という説明にとどまらず，プログラミングを行ううえでの作法・考え方にまで言及していることには感激すら覚えました。エラー退治というのはドラゴンを退治するのよりもはるかに難しく，時間のかかるものです。そういうものに対する心構えを，いろいろなケースを挙げて説明していただけるのはとても大切なことであるにもかかわらず，この手の連載ではあまり取り上げられることがないのです。そして1つひとつのステップを踏んで，わかりやすい短いプログラムからひとつずつ問題を解決してプログラムを次第に高度なものに完成していくのはプログラミングの基本ともいえると思います。フィルタを取り上げたのも正解だと思います。マシン語をやっていて初心者がいつも疑問に思うのは「こんな単純な命令セットだけでどうして複雑なことができるのだろう」ということです。そういう意味でも，難易度も内容もピッタリの選択だと思います。
湯澤聡（26）MZ-2500/2800，X1turboIII，CZ-600，PC-1360K　東京都

●泉氏の十八番，ついに万年暦が出ました。とてもわかりやすいと思います。「初心者向けとはいえ内容が簡単すぎる」とありますが決してそんなことはないと思います。初心者にとっては，このような記事はとても有り難いと思うし，それに自分もことX-BASICに関しては超初心者だからです。ハイレベルなユーザーにだってこの記事を読んで新たな発見や，再認識することがあるのではないでしょうか。
藤田康一（18）X68000PRO　静岡県

●うーむ，本気っぽいエディタだなあ。ページスクロールというのが"??"だが（いや，かえって使いものになるのかもしれないが），結構使えるものができるんじゃないだろうか。"^V"が2回以上押されたときにページスクロールの画面書き換えを中断して次のページを書きに行くというのは，なかなか本気でないとできない（やらない）ことだ。普通完成品が実用になるか否かは度外視して，とにかくエディタの基本構造なんかをプログラミングしてみるものだが，いきなり本気で作り始めるあたりはさすがに（？）祝社長なのであった。
小笠原陽介（21）PC-9801EX2，PC-E500　東京都

ごめんなさいのコーナー

10月号　ショートプロぱーてぃ
P.74　X1用リストの一部に誤りがありました。20700行のPRINT以下を，

```
PRINT STRING$(39, &H87);
```

に変更してください。

9月号　スーパーワイドコピー
P.53　リスト2に誤りがありました。131行のror.lをrol.lに，156～170行までの"beq"を"bgt"に変更してください。
また，作者名のところで，川上和彦さんの共同開発者である芦谷知二さんのお名前が落ちていました。お詫びいたします。

バグに関するお問い合わせは
☎03(230)7683(直通)
月～金曜日16：00～18：00

お問い合わせは原則として，本誌のバグ情報のみに限らせていただきます。入力法，操作法などはマニュアルをよくお読みください。
また，よくアドベンチャーゲームの解答を求めるお電話をいただきますが，本誌ではいっさいお答えできません。ご了承ください。

協力スタッフ募集 新しいOh!Xを君自身の手で

▼今月の特集「micro Computer入門」はいかがでしたか？　一見，ハード別情報誌の裏をかくような一般的な特集タイトルではありますが，マイクロコンピュータの世界をユーザー自身が身近に構築していくという意味でOh！Xらしい特集になったと思います。皆さんのご意見をお寄せください。

▼Oh！X編集部では，編集部員の内部異動に伴い，新しいメンバーでの再スタートを切ることになりました。新しいスタッフの個性が誌面にどう出るか楽しみですね。ちなみに，編集部のX68000所有率は相変わらず75%，西日本出身者率，ガラカメ全巻所有者率はともに75%をキープしているようです。

▼また，先月に引き続きOh！Xでは協力スタッフを募集します。Oh！Xで原稿を書いてみたい人，プログラム発表の場を作りたい人，出前の中華料理をともにしたい人，その他パーソナルコンピュータユーザーとしてさまざまな主張をお持ちの方をお待ちしています。応募の際は，住所・氏名・電話番号・略歴に加え，得意分野や自己PRおよび，過去1年間の特集をひとつ選び特集の主旨に即した内容の自由論文(感想ではなく)を6,000字程度にまとめて，Oh！X編集部スタッフ募集係まで郵送してください。

▼来月号は本誌がOh！MZからOh！Xに誌名を変更してちょうど2周年に当たります。特集ではC言語を取り上げ，できる限り基礎からの入門を行いたいと考えています。また，特別企画として，あっと驚く工作記事も用意していますのでご期待ください。

▼来年頭にはX68000が累計出荷台数で10万台突破が確実となってきました。10万台というのは，個人ユーザーがほとんどのX68000では極めて大きな意味を持ちます。たとえば，98が200万台といっても実際には2割ぐらいが個人のもので，あとはみんな企業による購入分です。要するにX68000の10万台は98の50～100万台に相当するといっても過言ではないわけです。ではまた，来月。

投稿応募要領

●原稿には，住所・氏名・年齢・職業・連絡先電話番号・機種・使用言語・必要な周辺機器・マイコン歴を明記してください。

●プログラムを投稿される方は，詳しい内容の説明，利用法，できればフローチャート，変数表，メモリマップ（マシン語の場合）に，参考文献を明記し，プログラムをセーブしたテープ（ディスケット）を添えてお送りください。また，掲載にあたっては，編集上の都合により加筆修正させていただくことがありますのでご了承ください。

●ハードの製作などを投稿される方は，詳しい内容の説明のほかに回路図，部品表，できれば実体配線図も添えてください。編集室で検討の上，製作したハードが必要な場合はご連絡いたします。

●投稿者のモラルとして，他誌との二重投稿，他機種用プログラムを単に移植したものは固くお断りいたします。

あて先

〒102 東京都千代田区九段南2-3-26井関ビル
日本ソフトバンク出版部
Oh!X「テーマ名」係

SHIFT・BREAK

▶「妹が，欲しいんだって？」とか「また，ふられたんだって？」とか，ボンカレーの田村正和はいやに子供の情報が速い。一体どうやって情報を収集しているのだろうか？　あの人の子供にだけはなりたくないなぁと，龍飛の銘水を飲みながら思う今日このごろ。P.S.真仁田君，あれは誤植だ。でも，人はいたかなぁ。敵がリガートみたいなのだよ。(H.U.)

▶10月号の特集を見たからではないのだが，最近ゲームにおけるゲーム性というものを考えている。パソコンに限らず，この世にゲームと名のつくものは，どのようにして人を楽しませているのか？　あるいは，どのように人は楽しむのか？　それが命題である。脳波を測定するときに，逆に電流を流したら，もしかしたら楽しくなれるだろうか？(亀)

▶学校の演習用のパソコンがPC-98VMからRAにかわった。スピードが速くなった。V30CPU＋RAM 容量640Kバイトも80386＋1.6Mバイトになった。でもやっぱりMS-DOSなんだよなぁ。MS-DOSが悪いとはいわないけどなんかもったいない。このDOSに386なんて必要なんだろうか，クロック数さえ速ければ関係ないのでは……。別にいいけど。(で)

▶この間，友達4人で榛名湖に行ってきました。さすがに高い場所にあるだけあって風が冷たく，ひと足早く秋の到来を感じました。その日はまるで少年(!?)のようにはしゃいでしまいました（ボートが漕げなくなっていた)。そんで，帰るときにけちって高速を使わなかったら，1時間の休憩をはさんで7時間もかかってしまった。(H.K.)

▶事態はそんなによくはない。愛の横断歩道がどうしたとか最年少の関取が誕生したとかそういった問題ではない。傘がないのだ。それさえも結果にすぎない。傘を必要とするに至った原因がどこかに潜んでいるはずなのだ。捜しに行くぞ。そして，脱出してやる。ビッグブラザーが好きな人は留まっていればいい。俺はトリックスターを育てる。(K)

▶おっと，スペースが6行しか余ってないぜ。そうそう，編集後記ってのは唯一原稿料にならない文章なわけですよ。っつうことは，特集のタイトルを『SHIFT BREAK』なんかにしちまったら，ライターにタダ働きをさせるというトリックが成立したり……するわけないか。(Mu)

▶K.S.さんは年貢を納めちゃうし，編集部は人事異動があるし，富士通は宮沢りえになるし，周りの環境が一変した。上司からは嫌いな英語の研修を受けさせられるし，仕事は急に忙しくなるし，疲れることの多い毎日だ。唯一の楽しみ，クリーミィマミの再放送の1話と2話の予約録画を立て続けに失敗して落ち込んでいる晩夏の夕暮れ。(KO)

▶今月号からこの編集部にお世話になることになりました。私の家にはストレス解消のためにツインファミコン，メガドライブ，CD-ROM²，X68000が存在しています。こんな私を人は“女・宮崎”と呼びます。でも実はPRINCESS²やKUSU-KUSUのライブに行き，渋谷で遊び回るというタダのミーハーなのでした。こんな私ですけど，これからもよろしくね。(E.O)

▶今月からOh！X編集部に配属されました。私自身はXユーザーではありませんが，X68000はバランスが取れたよいマシンなので，積極的に他のマシンのユーザーを啓蒙したいと考えております。ところで，日経バイト誌に書かれていた様に，5年後のX68000は，価格据置のままマルチメディア仕様の漢字対応UNIXマシンになるのでしょうか。(S)

▼神経質で人嫌い。ヨーロッパ俳優の中には，そうした役柄が実によく似合う人が多い。アパートメント・ゼロを観たときもつくづくそう思った。感心していると友人が一言。「あの主役と表情が似てるね」誰が？　私が？　ええ，どうせ暗い人間です。奥のほうの席って大好きだし。というわけで皆さんごきげんよう。(よ)

▶2台目のNV-V10000を買った。画質もさることながら，編集精度2フレームはダテじゃない。うまくやればきっちり1コマずつでも録画できる。「うなれ，AIさーぼぉ！　とっぺぇ100倍さーぁちい！」当分は遊べそうだ(あ，セレクタが足んねいや)。しかし，1台目はどこにいってしまったのでしょうね，松下さん。(ついに最年少脱出のU)

▶突然の話で申し訳ないのですが，9月をもって，よ嬢と一緒にOh！X編集部から新しいセクションへと移ることになりました。4年半も慣れ親しんできた編集部を離れるのは心苦しいものもあるけど，新しく若手スタッフを加えた新体制のOh！Xを引続きこれからも応援してくださいね。それでは，今度は別の誌面でお会いしましょう。(N)

▶久しく固定のメンバーでやってきたOh！Xだが，新雑誌の創刊に伴い，Nさんと(よ)さんが新しい編集部に移りました。かわって，入ってきたのが一見真面目なUNIXユーザーのSさんと，ゲームマシンマニア(？)のE.Oさんです。まさに対極的な2人の参加でOh！Xの今後の行く末が大いに注目されます。皆さん宜しくご指導くださいね。(T)

microOdyssey

消費税が実施される直前の3月末には，欲しいと思っていたものを一気に買いあさる人が大勢いた。結果として，本来なら買わないようなものまで買ってしまったり，重い本を買いすぎてタクシーで帰った（バカなやつ）など，余計な出費をしてしまった人も何人か知っている。逆に物品税が廃止になるのを待ってCDを買いあさったとかいう話はあまり聞かない。人間の出費に関する金銭感覚は結構あてにならないもののようだ。

さて，見直しだの廃止だのといわれる消費税だが，実際のところ，税の公平さとしてはどう受け止められているのだろう。よく消費者の声を代弁するかのように言われるのが，「日々の暮らしに欠かせない生鮮食料品にまで3％かかるのはよくない」とかいうものだ。皆さんはこれを公平な立場で聞いてまともな意見だと思われるだろうか？

たとえば，米価については「古米は安くなったが，新米は高くなった。私たちが本当に食べたいのは新米なのに」とぼやき，酒に関しては「一級酒以上は安くなったが，二級酒は高くなった。貧乏人に酒を飲むなと言うのか」とまたぼやく。どちらの声も気持ちはわかるが……。

生鮮食料品を非課税にすべきだという主張に対しては「では1本1万円のマツタケはどうなる？」といった話が出る。ほとんどの人にとって1本1万円のマツタケは贅沢品だからだ。ところが世の中には，キャベツより高い野菜は贅沢だと考える人もいるはずだ。そりゃ貧しい人なら……，と言うかもしれないが必ずしもそうではない。食事よりももっと違うことに使いたいと考えているだけのことだ。ついでに言うと，マツタケぐらいのものに課税か非課税かの境目があっても，実のところ多くの人にとってまったく関係がないのである。

また，経済力の乏しいお年寄りの声を代弁するかのように，税の負担をことさら強調する人もいる。この背景には老人に対する偏見や差別意識に近いものを感じる。老人なら誰でも古くなったテレビを見ながら白いご飯の食事を楽しみにしているとでも思っているのだろうか。

結果的には愚かな非課税品目の拡大は避けられるかなという気がしているが，それは制度上の別の事情によるもので，本質的に余計な考慮を払わないほうが税として公平なのだということを理解しての話ではないようだ。

そもそも，今の世の中で消費にかかわる価値観に一般性があると思うのは間違いだ。たとえば，年収200万円以下のアルバイターにとっても新米より新車のほうが贅沢とは限らない。まず，車，テレビ，オーディオときて，主食はカップラーメンだったりする。炊飯器を持っていない彼にはコシヒカリの値段など関係ない。

こういった生活態度を異常というならばしかたがない。しかし，異なる価値観を持つ人がこの国において平等であるならば，収入の同じ人が同じ額の消費を行った場合，税金もまた同額であるべきだと思う。夕食のおかずを一品へらしてAVテレビのクレジットにあてるかどうかは個人の自由であって，税額が変わるのは価値観に対する差別である。多様な価値観の存在する社会での公平さは何か？　もう一度考えなおしてみたい。　（T）

1989年12月号11月18日（土）発売

特集　C言語プログラミング入門

Oh!X 2周年特別企画

- 特大モニタプレゼント
- X68000にガイガーカウンタを接続する

X1用アクションゲーム　ACTIVE UNIT

Oh!X LIVE in '89

X68000用 Galaxy Forceより「Beyond the Galaxy」他

バックナンバー常備店

東京	神保町	三省堂神田本店5F	03(233)3312
	〃	書泉ブックマートBI	03(294)0011
	〃	書泉グランデ5F	03(295)0011
	秋葉原	T-ZONE 7Fブックゾーン	03(257)2660
	八重洲	八重洲ブックセンター3F	03(281)1811
	新宿	紀伊国屋書店本店	03(354)0131
	高田馬場	未来堂書店	03(200)9185
	渋谷	大盛堂書店	03(463)0511
	池袋	リブロ池袋店	03(981)0111
	〃	西武百貨店9F コンピュータ・フォーラム	03(981)0111
神奈川	横浜	有隣堂横浜駅西口店	045(311)6265
	〃	有隣堂ルミネ店	045(453)0811
	藤沢	有隣堂藤沢店	0466(26)1411
神奈川	厚木	有隣堂厚木店	0462(23)4111
	平塚	文教堂四の宮店	0463(54)2880
千葉	柏	新星堂カルチェ5	0471(64)8551
	船橋	リブロ船橋店	0474(25)0111
	〃	芳林堂書店津田沼店	0474(78)3737
	千葉	多田屋千葉セントラルプラザ店	0472(24)1333
埼玉	川越	黒田書店	0492(25)3138
	川口	岩渕書店	0482(52)2190
茨城	水戸	川又書店駅前店	0292(31)0102
大阪	北区	旭屋書店本店	06(313)1191
	都島区	駸々堂京橋店	06(353)2413
京都	中京区	オーム社書店	075(221)0280
愛知	名古屋	三省堂名古屋店	052(562)0077
	〃	パソコンΣ上前津店	052(251)8334
	刈谷	三洋堂書店刈谷店	0566(24)1134
長野	飯田	平安堂飯田店	0265(24)4545
北海道	室蘭	室蘭工業大学生協	0143(44)6060

定期購読のお知らせ

Oh!Xの定期購読をご希望の方は，とじ込みの振替用紙の「申込書」欄に何年何月号からをご記入のうえ，年間購読料6,720円（税込）を添えてお申し込みください。その際，裏面の通信欄に「○年○月号よりOh!X定期購読希望」と忘れずに明記してください。なお，すでに定期購読をご利用いただいている方には，購読期限終了と同時にご通知申し上げますので，同封の払込用紙をご利用ください。

海外送付ご希望の方へ

本誌の海外発送代理店，日本IPS（株）にお申し込みください。なお，購読料金は郵送方法，地域によって異なりますので，下記宛必ずお問い合わせください。

日本IPS株式会社
〒101 東京都千代田区飯田橋3-11-6
☎03(238)0700

11月号

■1989年11月1日発行　定価560円（本体544円）
■発行人　孫　正義
■編集人　橋本五郎
■発売元　（株）日本ソフトバンク
■出版事業部　〒102 東京都千代田区九段南2-3-26　井関ビル
Oh!X編集部　☎03(230)7681
出版営業部　☎03(230)7670 FAX 03(262)8397
広告営業部　☎03(230)7672
■印　刷　凸版印刷株式会社

バックナンバー案内

ここには1988年11月号から1989年10月号までをご紹介しました。現在1987年4, 1988年1,2,4～10, 1989年1～10月号までの在庫がございます。バックナンバーおよび定期購読のお申し込み方法については本文174ページを参照してください。

1988

11月号(品切れ)

特集 いまどきのプリンタ活用術
メカニズムを理解しよう/制御コード/文字と図形の混在印字/拡大文字のスムージング/外字登録ツール/S-H COPY/グラフィックのモノクロ出力/X68000のCOPYキー/オリジナル印刷キット/試用レポート
THE SOFTOUCH NEW PrintShop PRO-68K 他
OS-9/X68000入門(1) OS-9ってなに？
●STAR TREK for X68000
全機種共通システム シューティングゲームELFESⅣ

12月号(品切れ)

特集 パソコンはいま音楽の領域へ
なぜ自動作曲か/心地よい雑音の話/和音の読み方/美しい響きの要素/4分音符は歌い始める/古くて新しい音楽形式/FM音源の仕組み/Melody Box/MusicBASIC
●さよなら LIVE in '88 バッハ イタリア組曲他6本
●Oh!X 1周年記念特別企画「ちょっとあぶない福袋」
OS-9/X68000入門(2) OS-9のオペレーション環境
Z80マシン語ゲーム工房/C調言語講座PRO-68K
全機種共通システム ソースジェネレータSOURCERY

1989

1月号

特集 いきなり初春からハードウェア
デジタル回路入門/電子サイコロ/乱数発生器/X1turboバンクメモリ拡張/X68000用CP/M-80システム 他
1988年度GAME OF THE YEAR ノミネート作品発表
●MZ-2500用 Hyper Game Book
●LIVE in '89 エンデューローレーサー/アルルの女
●ようこそ,セガ・メガドライブ!!
C調言語講座PRO-68K/Z80マシン語ゲーム工房
全機種共通システム パズルゲーム LAST ONE/FLICK

2月号

特集 マシン語"でじたるざんまい"
アーキテクチャからのマシン語入門/アセンブラへの招待/超入門Z80マシン語活用術/X68000料理教室
THE SOFTOUCH 彩CRONE/Final Ver.3.2 他
●X1/X1turbo用RPG FLAME
Z80マシン語ゲーム工房 最終回 爆発,そして完成へ
C調言語講座PRO-68K (8) とおりゃんせなのである
OS-9/X68000入門(3) ついに発売！OS-9/X68000
全機種共通システム 高速エディタアセンブラREDA

3月号

特集 BASIC"おもちゃ箱"
ピコピコゲームから重力シミュレーションまで
●X1/X1turboでMZ-700用スペハリ/ロボットゲームTAMA
●数値演算を高速化 FLOAT2+.X
OS-9/X68000入門(4) C言語の概要を見る
C調言語講座PRO-68K(9) ニホン語, 不得意
新連載予告編X68000マシン語プログラミング入門
全機種共通システム 浮動小数点演算パッケージSOROBAN
THE SOFTOUCH/LIVE in'89/知能機械概論/猫とコンピュータ

4月号

特集 ゲーマーたちの"新深夜族"宣言
1988年度GAME OF THE YEAR
新連載 X68000マシン語プログラミング
●X1/X1turbo用パズルゲーム ロボット衛兵
●MZ-700用ゲームパッケージ System-7B
●LIVE グラディウスII/ザ・スキーム/パワードリフト
連載 C調言語講座PRO-68K/OS-9/X68000入門
全機種共通システム SLANG用実数演算ライブラリ
特別付録 X68000イメージCGポスター

5月号

特集 MIDIサウンドデータ料理術
LA音源をFM音源でシミュレート/X-BASICでMIDI制御
特別企画 第4回「言わせてくれなくちゃだワ」
●シャープパソコンフォーラム'89 in赤坂
●詳解Human68k ver.2.0
●MZ-2500, X1/X1turbo用 戦略的ライトサイクルゲーム
連載 C調言語講座PRO-68K/ OS-9/X68000入門
X68000マシン語プログラミング
全機種共通システム ソースジェネレータRING

6月号

特集 これからのXfamily
X68000に光磁気ディスクを/学習リモコンの製作
THE SOFTOUCH ライトニングバッカス/Might and MagicⅡ他
●OPMA用外部関数による KENBAN.BAS
●X1/X1turbo用ドライブゲーム Spirit of Rally
●X1turboZ用 これ,パズルなんですか。
MZ-2500 MIDI入門(1)MIDIボードを作る
C調言語講座PRO-68K/X68000マシン語プログラミング
全機種共通システム 超小型コンパイラTTC

7月号

特集 3Dグラフィックへの飛翔
Zバッファアルゴリズム/スムースシェイディング 他
THE SOFTOUCH Terazzo PRO-68K/アドヴァンスト・ファンタジアン
新連載
DōGA・CGアニメーション講座
MZ-2500用グラフィックエディタ作成講座
マシン語カクテル in Z80's Bar
X-BASICプログラミング調理実習
全機種共通システム TTC用パズルゲームTIC BAN
X68000マシン語プログラミング/C調言語講座PRO-68K 他

8月号

特集1 X1プログラミングガイドブック
PCGの基礎から奥義まで/超高速ラインルーチン 他
特集2 3Dグラフィックの深淵へ
スキャンラインZバッファ/3Dモデリング 他
新連載 (で)のショートプロぱーてぃ
X68000マシン語プログラミング/C調言語講座 PRO-68K
X-BASICプログラミング調理実習/DōGA・CGA講座
MZ-2500用グラフィックエディタ/Z80's Bar 他
全機種共通システム CP/M用ファイルコンバータ

9月号

特集 活用ハードディスク&プリンタ
各社ハードディスク接続総チェック/ハードディスク雑学講座/COPYキーメニュー/ビデオプリンタ活用プログラム 他
THE SOFTOUCH ジェノサイド/琉球/mFORTH Compiler
●サイバースティックで遊ぶ 不思議な環境ソフトの世界
●X1/X1turbo用シューティングゲーム Defeat X
ショートプロぱーてぃ/MZ-2500グラフィックエディタ 他
[X68000] X-BASIC/マシン語/C調言語講座/DōGA・CGA
全機種共通システム 生物進化シミュレーションBUGS

10月号

特集 ゲーム面白心理学
ソーサリアン・宇宙からの訪問者/ファンタジーゾーン ねじ式/ガウディ・バルセロナの風/サバッシュ 他
●MZ-700用シューティングゲームSide Roll-F
●X1/X1turdo用カードゲームBonding
ショートプロ/Z80's Bar/MZ-2500グラフィックエディタ
X68000マシン語/X-BASIC/C調言語講座/DōGA・CGA
THE SOFTOUCH Z'sTRIPHONY DIGITAL CRAFT/James68K
全機種共通システム 小型インタプリタ言語TTI

ソフト名 **音感 ONKAN**
対照機種 **X68000系列**
金額 **たったの4千円**(内税)

〒771-15
徳島県板野郡土成町
吉田字御所屋敷
TEL.(0886)95-2098

一言で言うと音痴をなおすソフトです。私の会社では音声認識の研究をしていますが、その過程で不思議なことを発見しました。多くの人が音痴であり、かつ音痴であることを知らないということです。

しかし考えてみると無理もありません。楽器であれば客観的に聴き比べ、音程を調整しつつ練習することが出来ます。しかし、肉声は、特に自分の声というのは、あまりにも日常耳にしてる故に逆に自分の声がどういう音程であるか知ることが出来ず、練習が出来ないのです。いくらカラオケでがなってみても、音程の練習になりません。

そう、今まで自分の出してる音程を知る機械というのはありませんでした。ギターなんかのチューニングメーターというのはありましたが、適用出来る音域というのがとても狭く、肉声のように複雑で、音域の広いものには使えません。音程を知るなんて簡単そうに思えるでしょうが、たとえ単音であっても肉声の音程を認識するのはとても難しい原理が必要で、なかなか大変だったのです。

その大変なことを実現したのが「音感」というソフトです。といっても操作はX68000にマイクをつなぎ、歌えば、声に合わせて画面の鍵盤の位置に音階が表示されるだけの簡単なものです。その鍵盤の位置を見ながら自分の声を調整すれば、秘かに音痴をなおしてしまえる訳ですね。おおくの人は、これによって半音どころか2音くらいずれて平気で歌ってたことを反省させられるでしょう。

ようするにそれだけのソフトで操作も簡単なのです。これだけの機能だとしても4000円は適値ですが、さらにオマケが色々とついています。

まず、歌った音階を120秒覚えていて、修正することが出来、しかも、それをBASICのMML形式に変換出来ます。常駐させたなら、ソフトキーボードからの入力と同じように動作します。こう書くととても難しいことのようですが、お使いになってみればファイルとかのことを何も考えなくてよくまるで空気のように利用出来ることに気づくでしょう。

当然、画面のキーボードでも演奏出来るしそれをMMLにすることも出来ます。「音感」はFM音源を使ってませんからFM音源の伴奏で歌いながら音痴の特訓をするなんてことも出来ます。マイクの代わりにAUDIOINにキーボードをつなげば、キーボードからの音をMML変換することだって出来てしまいます。

購入方法は少し普通とは違います。

あなたにはまず、4千円を支払って頂きます。郵便局で郵便コガワセ4千円分を買い不透明な封書に自分の住所氏名と「音感希望」を明記して送って下さい。これが最も安価な送金方法です。安全性を重視したい方は現金封筒で送って下さい。送料がかかりますが確実です。

こちらからお送り致しますのはお買い頂く「音感」の他、デモ用のマイクアンプと、パラサイトウエアの原本と生フロッピー、返信用切手が入っています。このマイクアンプは音感の性能をすぐその場で確認して頂く為のもので商品ではありません。「音感」の使用はテープレコーダなどのアンプ機能を使って利用出来ます。音感のデモをご覧頂いた後、このマイクアンプは原本と一緒に送り返して下さい。というのはこのアンプは基板そのままで商品の体裁をしていないからです。

(どうしても必要であれば1500円頂きます)

「音感」の商品範囲は、フロッピー1枚+説明書までです。それ以外のものは1週間以内に送り返して下さい。必要な切手は貼ってありますから郵便ポストに入れるだけです。

パラサイトウエアとは「音感」というソフトのパッケージに宿借りする形を取るソフトの新しい販売形態です。パラサイトとは寄生という意味で、「音感」のパッケージに寄生し、流通コストをかけずに流通しようという試みです。

「音感」はその商品の第1号です。ですからパラサイトウエアの方には、現在まだあまり商品は入っておりません。現在1件につき100円から500円程度のソフトがいくつか入っています。全部買っても5千円にはなりません。(9月8日現在) 今あるのは超高速なマンデルブロ描画ソフトやプログラム作成時に便利なツール、ピコピコゲーム等が入っています。

これらはひやかしで結構ですからメニューだけでもご覧下さい。もし気に入ったらものがあればお買い下さい。マウス操作で簡単に選べるようになっており、購入する時は圧縮されてる原本のファイルを添付の生ディスクに自動的にコピーするようになっています。ですからすべてお買いになる場合でも原本は返して頂きます。というのは原本の方にお買いになったかどうかの情報が入っているからです。

パラサイトウエアの商品をお買いになった方は、原本を返送頂く時に、郵便コガワセ等で不足額をお支払い下さい。

音感そのものがデモの段階で気に入らない場合は、音感のデモの終了後、購入を選択せずそのまま送り返して下さい。手数料を1500円頂きますから差額の2500円を返却させて頂きます。

こちらとしては4000円なら安すぎると思ってるのですが、上のような事情で「音感」を一般の販売店を通すことが出来ない為、このサービスをさせて頂きます。

音感のソースリストもパラサイトウエアとして3000円で別売になっています。(パラサイト商品で最も高額) ライバル会社の方に音感を逆アセンブラにかけて解析する手間を省いて頂こうという優しい配慮です。こちらの方には「音感」のソースを使ってBASICで簡単にADPCMのPCMへの復元や超高速の波形表示機能などが出来る拡張関数機能もついています。「音感」のMML変換ルーチンは実行時リンク方式ですから標準以外のMMLにも対応出来ます。楽譜にするにも難しくはないでしょう。その筋の方もどうぞ。

あなたのお作りになったソフト。100円以上の価値があるとお思いでしたら、「音感」の中にパラサイトウエア応募キットが無料でありますからそれに従って応募して下さい。売上4万円までの手数料は25%しか頂いていません。とてもお得です。たとえば200円のピコピコゲームでも70人に売れれば1万円です。ちょっとした小遣い稼ぎになるでしょう。内容は問いません。ただし、他人の著作権を犯すものとかウイルスなど迷惑ソフトは困ります。詳しくは応募キットにて。

また応募キットには全部アセンブラソースリストがついています。結構便利なツールもあります。お勉強をしたい方などこれだけでもお得です。

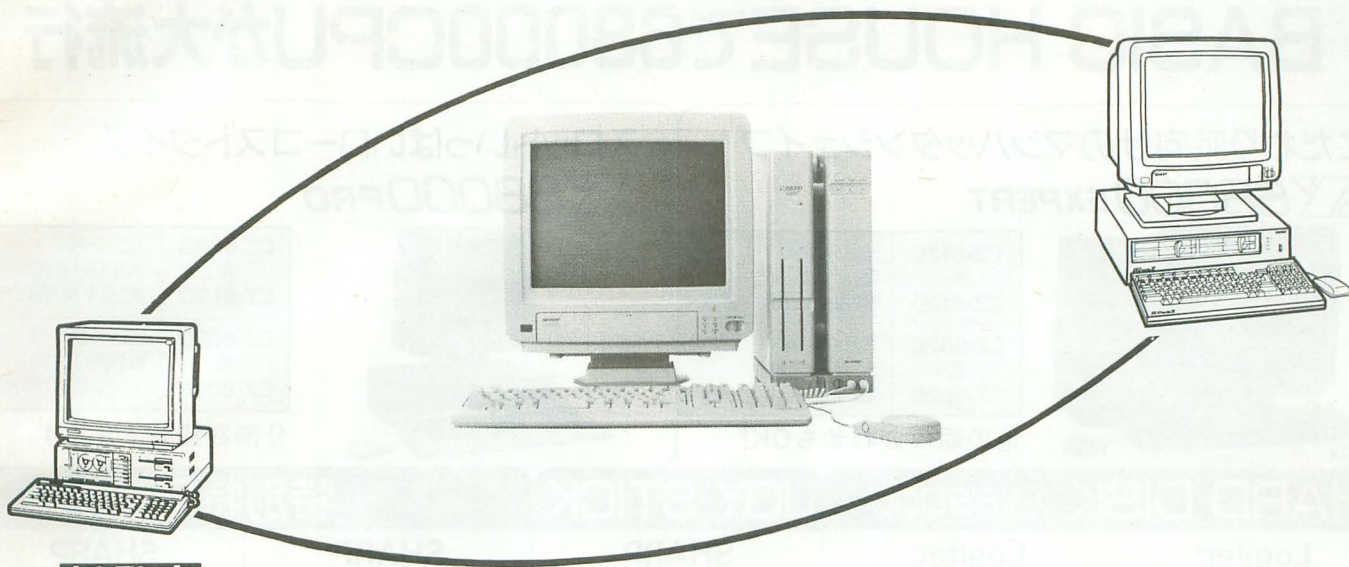
待機時は色々な機能が使えます

実時間で音階表示をしながら波形まで見えます

私どもの会社は田舎にあります。田舎ではソフトといっても都会のように販売店に行けば在庫があってすぐ買えるという訳にはいきません。パッケージを見て買えるというのもマレです。パラサイトウエアというのはそのようなみんなの不満を背景に生まれました。メイン商品に同居し、いわば小さなお店を貴方の家の中に開かせて頂こうというものです。これならどんな田舎の人も不便は感じません。そして都会の人にだって安価というメリットを獲られます。自分の会社のソフトだけでなく、広く一般のソフトを作れる方々にも利用してもらいたいというのが願いであります。

こんなことを不法コピーユーザー揃いの98系列でする程の勇気もありませんので、X68000で始めさせて頂きました。いまのところ原本にプロテクトはかけていません。みなさまの御賛同を願っております。

外国製のMS-DOSにもアクセス出来る!

新発売

X68000用

SUPER DEVICE MONITOR "T"

X1turboや、MZ-2500ではもうお馴染の『SUPER DEVICE MONITOR "T"』のX68000用がいよいよ発売されました。

今までは、手探りで行なっていたプログラムの開発が、容易に出来る様に成ります。
例えばCコンパイラーや機械語を使ってソフトを自作している場合、1バイトの定数等を書き換えるのにいちいちエディターでソースプログラムを書き直してからアセンブルからもう一度やり直さなければ成らなかった作業が『SUPER DEVICE MONITOR "T"』を使うと1バイト単位で書き換えられるので簡単に出来る様に成ります。特にハンドアセンブルをする方には今までに無かった快適な開発環境を提供します。

★アクセスしたセクターは、縦横チェックサム付で表示して、ワープロ感覚で変更・複写・スクロール等の多彩なエディット機能が1バイト単位で使えます。

★S-RAMやIPLなど通常アクセス出来ない部分を含めてX68000内で呼び出せるメモリーは殆ど総てセクター単位でアクセス出来ます。

★RS-232Cを使うと任意のボーレートでX68000同士は勿論、他機種にはその機種用の『SUPER DEVICE MONITOR "T"』を介して、特殊なデータ圧縮法により、最高速では通常の32倍(理論値)の超高速で転送が行ええます。例えばフォーマットしたばかりの2Dのディスク1枚分を1200ボーで転送すると約8分間で転送が出来ます。
(X1のみ不可)

★256バイトを1セクターとしIPL-ROM、S-RAM、MIN-RAMなどが別々のディバイスとしてアクセス出来ます。

★X68000標準フォーマット以外のフォーマットもアクセス出来る可変フォーマット機能付です。

★RS-232Cのボーレートの変更は、ボタン1つで簡単に出来ます。

X68000用のみ最高1300ガウスの磁気を浴びても大事なフロッピーディスクが安全に守られる、三菱鉛筆製の磁気遮蔽機能付『uniフロッピーディスクケース』に入っています。

SUPER DEVICE MONITOR "T"

X68000	5"	2HD	15,000円
X1	5"	2D	10,000円
X1turbo(2HDは受注生産)	5"	2D/2HD	13,000円
MZ-2500・MZ-2800	3.5"	2DD	13,000円

*MS-DOSはマイクロソフト社の商標です。
*商品の価格には消費税は含まれていません。

▶お求めは全国の有名マイコンショップでどうぞ。
通信販売をご希望の方は当社へ直接、商品名・機種名・メディア名・住所氏名・電話番号を明記の上、現金書留にてお申し込みください。(送料無料)

BLUE SKY Co.

株式会社 BLUE SKY
〒411 静岡県三島市加茂16-4
☎0559-72-6710

信用と実績を誇る BASIC HOUSE

北関東最大の68000専門店

SHARP SONY NEC YHP 横河ヒューレットパッカード Apple Computer

BASIC HOUSEで68000CPUが大流行

こだわり派向けのマンハッタンシェイプ X68000 EXPERT

CZ-612C & CZ-612D	36回分割例 第1回¥18,900 第2回¥16,400×35回
CZ-602C & CZ-602D	36回分割例 第1回¥13,400 第2回¥13,100×35回

他の組み合わせもOK!

スロットいっぱいローコストタイプ X68000 PRO

CZ-662C & CZ-612D	36回分割 第1回¥17,900 第2回¥14,700×35回
CZ-652C & CZ-603D	36回分割 第1回¥12,700 第2回¥10,700×35回

分割数変更できます

HARD DISK X68000 / JOY STICK / 特別特価

Logitec	Logitec	SHARP	SHARP	SHARP
LHD-34V	LHD-32V	Cyber Stick	CZ-620H	CZ-611CGY
~~¥158,000~~→¥134,000	~~¥128,000~~→¥110,000	~~¥23,800~~→超特価	~~¥178,000~~→¥89,800	~~¥398,000~~→¥298,000
容量40Mバイト	容量20Mバイト		容量20Mバイト	
オートシッピング機能 PC98シリーズ用ですがX68000でも動作します。	オートシッピング機能 PC98シリーズ用ですがX68000でも動作します。	無段階変更連射機能/多数のトリガ/127段階全方向取り込みの Joy Stick	HDD内蔵モデル(ACEHD/PROHD/EXPERTHD)の方もご相談下さい。	ACEHDのグレーモデル/Human68k Ver. 2も付いてこの値段!!台数少なし至急CALL!!

PRINTER X1/X1 turbo/X68000 / SCANNER

SHARP	SHARP	EPSON	EPSON	EPSON	OMRON	SHARP
CZ8PC3	CZ8PC4	VP1000	VP135EX	HG3000	HS10RII	CZ8NS1
~~¥65,800~~	~~¥99,800~~	超特価	超特価	~~¥246,000~~	~~¥49,800~~	~~¥185,000~~
24ドット80桁熱転写カラー漢字プリンタ	48ドット80桁熱転写カラー漢字プリンタ	24ドット136桁プリンタESC/Psuper機能内蔵単票連続両用紙標準サポート高速220字/秒(ドラフト)	24ドット/136桁プリンタESC/Psuper機能内蔵単票連続両用紙標準サポート	24ドット/136桁インクジットプリンタ 低雑音 45dB 漢字220字ANK550字(ドラフトモード)	低価格、多機能、白黒ハンディスキャナバーコードリード機能付き	フルカラー/A4サイズ/200DPI/専用パラレルボード(別売り)の使用により高速取り込みも可能

Come on New BASIC House

いよいよ軌道に乗ってきた大田原営業所

野崎街道沿い美原公園野球場裏、ダイユー隣白い建物です。よろしくお願いします。

TEL.0287-23-5352

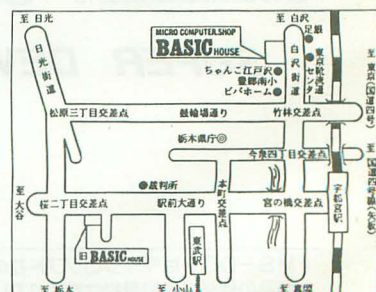
大田原営業所では通信販売は扱っておりません。通販ご希望の方は宇都宮本店にお願いします。

通販御希望の方は購入品名、住所、氏名、電話番号を書いた紙と(代金+送料¥1,000)× 消費税1.03を同封して現金書留でお申し込み下さい。釣銭は無いようお願いします。

全国通販OK!

- 低金利クレジットあつかっております。
- 支払方法は相談に応じます。
- 商品組み合わせもご相談下さい。

表示価格は特に明記されている場合を除き消費税は含まれておりません。

MICRO COMPUTER.SHOP BASIC HOUSE

全国どこでも発送可　長期クレジットOK　送料全国均一¥1,000　宅配便にて即日配送

株式会社計測技研　本社営業部/マイコンショップ/通販部 〒321 宇都宮市竹林町503-1 TEL0286-22-9811 FAX0286-25-3970

マイコンショップ BASIC HOUSE お申し込み・お問い合せは ☎0286-22-9811(代)

燃えるぜ！ 新製品だ！

Co-Processor & Ext. RAM(max 4M) on One Board

KGB-X68PRK

数値演算プロセッサと増設メモリを1枚のボードに収納

マンハッタンシェイプはスロットが少ないと嘆く君に計測技研が贈るおいしいアイテム
SHARP純製品の2M/4M増設メモリと数値演算プロセッサボードとコンパチブル
最小構成：RAM 512Kbyte コプロセッサ無し
最大構成：RAM 4Mbyte コプロセッサ付き
コプロセッサ、RAMは後から増設することも可能！

注意） 初期型/ACE/PROの場合、専用1MバイトRAMを増設してメインメモリを2Mバイト以上にしている必要があります。

64180CPU on X68000　Mach 180

★Z80/HD64180のプログラムをX68000上で開発できる！

- CPUにHD64180（クロック10MHz/ノーウェイト）を採用、8ビット最高速
- メモリーは64kバイトを実装、64k CP/Mとして使用可能
- CP/M-80 BDOSエミュレータの使用によりBDOSレベルでのCP/M-80互換を実現
- Human68kのコマンドと同一ディスク上での混在使用が可能
- モード切り替えの必要なし
- CP/Mディスクドライバによりturbo CP/M（2HD）のフロッピーの直接アクセスが可能

BASIC HOUSE オリジナルハードウェア

KGB-X68ADC	KGB-X68PIO	KGB-PIO	KGB-AD12	KGB-X1S	Melody Box
高速A/D変換	高絶縁パラレルI/O	高絶縁パラレルI/O	高速A/D変換	汎用アナログ入力 デジタル入出力	MIDIインタフェースユニット
¥128,000	¥68,000	¥42,000	¥118,000	¥19,800	¥16,800
マルチレンジ/12bit/5μsec変換/シングルエンド16ch又は差動入力8ch/アセンブラ/BASIC/Cのサンプルソフト付き。	フォトカプラ外部電源供給による高絶縁16bit入出力/アセンブラ/BASIC/C/のサンプルソフト付き。	フォトカプラ外部電源供給による高絶縁16bit入出力。	マルチレンジ/12bit/5μsec変換/シングルエンド16ch又は差動入力8ch選択可能。	8bit/8chのA/D変換8255による24bit入出力ボード/フリースペース付き。	X68000にMIDI機器を接続するためのユニットです/RS232Cボードに接続/各種ドライブソフト付属。
X68000用	X68000用	X1用	X1用	X1用	

BASIC HOUSE X68000 オリジナルソフトウェア

B6-6301	B6-6302	B6-6303	B6-6305	B6-6306	B6-6307
BASIC拡張関数パッケージ	CP/M 68K エミュレータ	アイコンエディタ	C言語ライブラリ	BASIC拡張関数パッケージ（C言語ライブラリ付）	Toys & Tools
¥9,800	¥19,800	¥4,800	¥6,800	¥14,800	¥6,800
なんでこんな関数がないんだ！ X-BASICの機能をアップさせる約50種類のパッケージ	CP/M 68KのBDOSコール機能をエミュレートし、CP/M 68KのアプリケーションをHuman68kで実行。	オリジナルアイコンを作ろう！ビジュアルシェルで使用するアイコンを登録変更。	XBASTOCが通らないBASIC拡張関数パッケージをXBASTOCで利用するためのC言語用ライブラリ。	やっぱり両方あったほうがいい！ お得なB6-6301とB6-6305のセット。	コマンドは多いに越したことはない。使って楽しく、しかも便利な外部コマンドをパッケージ化。

全国どこでも発送可　長期クレジットＯＫ　送料全国均一¥1,000　宅配便にて即日配送

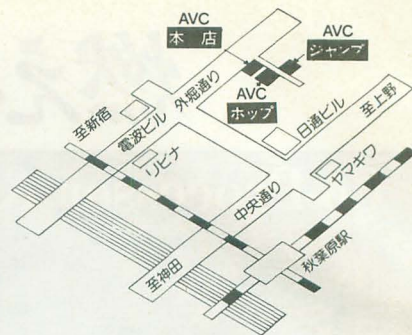
株式会社計測技研　本社営業部／マイコンショップ／通販部　宇都宮市竹林町503−1　TEL0286-22-9811　FAX0286-25-3970

マイコンショップ BASIC HOUSE　お申し込み・お問い合せは ☎0286-22-9811（代）

秋葉原価格でローンができます
電気の街秋葉原で 25年の信用!!

AVCフタバ

☎03(253)7661

〒101 東京都千代田区外神田3-2-3 ☎03-253-7611(代)

今すぐ もよりの電話から	仙 台 022-264-3704	名古屋 052-452-3271	広 島 082-295-6873
札 幌 011-611-5104	新 潟 0252-75-4175	大 阪 06-311-3931	福 岡 092-481-2494

AVC本店 / AVCジャンプ / AVCホップ / 至新宿 / 電波ビル / 外堀通り / 日通ビル / 至上野 / リビナ / ヤマギワ / 中央通り / 至神田 / 秋葉原駅

X68000の情報のすべて!(当店はX68000の認定代理店です。お気軽にご相談下さい)

X68000 待望の新しい仲間登場!!

X68000 PERSONAL WORKSTATION EXPERT・EXPERT HD

集積度を高めた"マンハッタンシェイプ"2Mバイトのメインメモリを標準実装、Human68K ver2.0搭載(CZ-602C) 更に40MBのHDDを搭載(CZ-612C)あくまでもX68Kにこだわるマシン。

〔写真のモニタは別売です〕

CZ-602C 標準価格¥356,000
CZ-612C 標準価格¥466,000

AVC特価

X68000 PERSONAL WORKSTATION PRO・PRO HD

拡張I/Oスロットを4スロット標準装備、メインメモリ1MB、Human68K ver2.0搭載(CZ-652C) 更に40MBのHDDを搭載(CZ-662C) 新しいX68Kの発見があるはずだ。

〔写真のモニタは別売です。〕

CZ-652C 標準価格¥298,000
CZ-662C 標準価格¥408,000

AVC特価

X68000 PERSONAL WORKSTATION ACE・ACE HD

在庫有限

従来機も忘れずに!!

CZ-611C(HDDタイプ)
¥399,800
➡AVCフタバ特価
〔写真のモニタは別売です〕

お勧めディスプレイコーナー 組合せは自由、価格はお気軽にご相談下さい。

CZ-612D
標準価格¥118,800
AVC特価
- 0.31mmドットピッチ
- TVチューナ搭載
- 3モードオートスキャン
- チルト台同梱

CZ-603D
標準価格¥84,800
AVC特価
- 0.31mmドットピッチ
- TVチューナ無し
- 3モードオートスキャン
- チルト台同梱

CZ-602D
標準価格¥99,800
AVC特価
- 0.39mmドットピッチ
- TVチューナ搭載
- 3モードオートスキャン
- チルト台同梱

CU-21CD
標準価格¥139,800
AVC特価
- 0.52mmドットピッチ
- TVチューナ無し
- 3モードオートスキャン
- チルト台取付不可

型番	品名	標準価格	販売価格
CU-14BD	ディスプレイ	¥ 64,800	AVCフタバ特価
CU-14FD	ディスプレイ	¥ 84,800	AVCフタバ特価
CU-14GD	ディスプレイ	¥ 69,800	AVCフタバ特価
CZ-860D	ディスプレイ	¥ 99,800	AVCフタバ特価
CZ-830D	ディスプレイ	¥ 90,600	AVCフタバ特価
DZ-880D	ディスプレイ	¥102,100	AVCフタバ特価
BF-68PRO	CRTフィルター	¥ 19,800	AVCフタバ特価
CZ-502F	FDD(2DD)	¥ 99,800	AVCフタバ特価
CZ-503F	FDD(2D)	¥ 49,800	AVCフタバ特価
CZ-6BE1A	1MB (増設RAMボード)	¥ 38,000	AVCフタバ特価
CZ-6BE2	2MB (増設RAMボード)	¥ 79,800	AVCフタバ特価
CZ-6BE4	4MB (増設RAMボード)	¥138,000	AVCフタバ特価
AN-160SP	アンプ内蔵スピーカー	¥ 59,800	AVCフタバ特価
CZ-8BS1	FM音源ボード	¥ 23,800	AVCフタバ特価
CZ-6BN1	スキャナ用パラレルボード	¥ 29,800	AVCフタバ特価

型番	品名	標準価格	販売価格
CZ-8PC2	熱転写プリンタ(24ドット)	¥ 69,800	AVCフタバ特価
CZ-8PC3	熱転写プリンタ(24ドット)	¥ 65,800	AVCフタバ特価
CZ-8PC4	熱転写プリンタ(48ドット)	¥ 99,800	AVCフタバ特価
AN-8TU	RGBシステムチューナ	¥ 33,100	AVCフタバ特価
CZ-8PK7	プリンタ(80桁)	¥122,000	AVCフタバ特価
CZ-8PK8	プリンタ(136桁)	¥152,000	AVCフタバ特価
CZ-8PK9	プリンタ(80桁)	¥ 89,800	AVCフタバ特価
CZ-6VT1	カラーイメージユニット	¥ 69,800	AVCフタバ特価
CZ-8BV2	カラーイメージボード	¥ 39,800	AVCフタバ特価
CZ-6BU1	ユニバーサルI/Oボード	¥ 39,800	AVCフタバ特価
CZ-6BG1	GP-IBボード	¥ 59,800	AVCフタバ特価
CZ-8TM1	モデム	¥ 29,800	AVCフタバ特価
CZ-8TM2	モデム	¥ 49,800	AVCフタバ特価
CZ-8NT1	トラックボール	¥ 13,800	AVCフタバ特価
CZ-6SD1	システムラック	¥ 44,800	AVCフタバ特価

型番	品名	標準価格	販売価格
CZ-6BF1	増設RS232Cボード	¥ 49,800	AVCフタバ特価
CZ-6BP1	数値プロセッサボード	¥ 79,800	AVCフタバ特価
CZ-6EB1	I/Oボックス	¥ [illegible]8,000	AVCフタバ特価
CZ-234LS	AI開発ツール	¥188,000	AVCフタバ特価
CZ-219SS	OS-9	¥ 29,800	AVCフタバ特価
CZ-227BS	TOP財務会計	¥200,000	AVCフタバ特価
CZ-213MS	MUSIC PRO-68K	¥ 18,800	AVCフタバ特価
CZ-214MS	SOUND PRO-68K	¥ 15,800	AVCフタバ特価
CZ-212BS	ビジネス PRO-68K	¥ 68,000	AVCフタバ特価
CZ-211LS	Cコンパイラ PRO-68K	¥ 39,800	AVCフタバ特価
CZ-141SF	NEW-Z BASIC	¥ 18,800	AVCフタバ特価
CZ-137SF	turboZ's STAFF	¥ 19,800	AVCフタバ特価
CZ-133SF	モデムターミナルソフト	¥ 25,800	AVCフタバ特価
	Z'STAFF PRO-68K	¥ 58,000	AVCフタバ特価
	kamikaze	¥ 68,000	AVCフタバ特価

CZ-8NJ2

アナログジョイステック
標準価格 ¥23,800

AVC特価¥???

X1turboZIII

X1ターボシリーズの独自の機能を全継承。VCCIゼロdB基準に適合させた。

CZ-888C…¥169,800
CZ-860D…¥ 99,800
合計………¥269,600

特価 ???

応談 価格はご相談に応じます、電話でお問い合せ下さい。

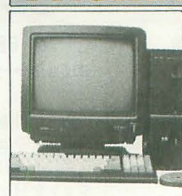
X1turboZII

X1turboZの本格派セット。TV付2モードオートスキャンディスプレイ。

CZ-881C…¥179,800
CZ-880D…¥109,800
合計………¥289,600

特価 ???

応談 価格はご相談に応じます、電話でお問い合せ下さい。

X1twin

HEシステムを搭載、最上級ゲーム機とパソコンが合体。

CZ-830C…¥ 90,800
CZ-830D…¥ 90,600
合計………¥190,400

AVC特価¥???

●但し消費税(3%)は別途請求させていただきます。●分割回数は3回~48回まで自由に選べます。

●セットの組合せは自由!広告に出ていない他の機種はお問合せ下さい。

●頭金なし(手軽な電話クレジット)●製品先取り(お支払いは約1~2ヶ月後から)●低金利クレジット(1回の支払いは2,700円以上で3~48回。ボーナス併用も可)●カレッジクレジット(保証人なし。但し満20歳以上の学生の方)●18歳未満の方(ご両親が代理購入者としてお申し込み下さい)
●納期(通常の場合、当社に申込書が到着後1週間以内。特に人気のある商品で品薄の場合、少々納期が遅れることがありますので御了承下さい)
●完全保証(すべてメーカー保証書付。アフターケア万全)●全国代引(お届けした者に、代金をお支払いいただく方法です。但し手数料1,000円)

AM10時からPM7時まで受付 日曜・祝日も営業

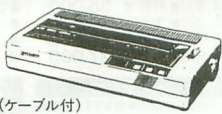
“プリンタ・コピー・ファクス”1台3役のスグレモノ

パソコンファクス「MZ-1V01」限定セット販売！

- ●MZ25セット（インターフェース ソフト付）
  標準価格合計￥342,800を **￥168,000**
- ●MZ28セット（インターフェース ソフト付）
  標準価格合計￥377,800を **￥198,000**
- ●PC98セット（インターフェース ソフト付）
  標準価格合計￥377,800を **￥198,000**
- ●MZ-1V01本体のみ
  標準価格￥278,000を **￥120,000**

※上記セットをご注文の際は3.5か5インチのご指定をしてください。

新製品！ ハガキもOK、New MZプリンタ

漢字カラー熱転写プリンタ
「シャープMZ-1P22」

標準価格￥59,800 **特価￥38,640**（ケーブル付）

〈24×24ドット漢字・7色カラー・漢字30字/秒高速印字・MZ1P17とフルコンパチ・5KBのバッハメモリ付〉
適応パソコン→MZ2000、2500、5500、6500シリーズ、X1シリーズ、X68000シリーズ他。

X68000大特価！クレジットOK

●X68000EXPERT
(CZ-602C)
1MB/FDD×2
定価￥356,000
《クレジット大特価》
月々￥9,400×36回

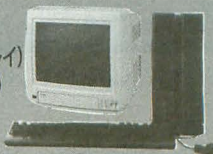
●X68000PRO
(CZ-652C)
1MB/FDD×2
定価￥298,000
《クレジット大特価》
月々￥7,900×36回

●X68000EXPERT-HD
(CZ-612C)
1MB/FDD×2、
40MB/HDD×1
定価￥466,000
《クレジット大特価》
月々￥12,300×36回

●X68000PRO-HD
(CZ-662C)
1MB/FDD×2、
1MB/HDD×1
定価￥408,000
《クレジット大特価》
月々￥10,800×36回

AIBIT
アイビット電子株式会社

X68000激安大特価セット！

■CZ-611C(本体)
■CZ-611D(ディスプレイ)
■CZ-1P22(プリンタ)

定価合計￥600,400を
特価￥388,350

●モデムCZ-8TM1(ソフト付)をプレゼント！

富士通FM-TOWNSセット大特価ご奉仕!!

Aセット ①本体/FMTOWNS-1②CRT/FMT-DP531③キーボード/FMT-KB101④OS/TOWNSシステムソフトウェアーV1.1⑤本体増設/内蔵マイクロFDドライブ⑥OS/MS-DOSエミュレータV1.1

①〜⑥計　標準価格￥478,000
ご奉仕大特価￥398,000

Bセット ①本体/FMTOWNS-2②CRT/FMT-DP531③キーボード/FMT-KB101④OS/TOWNSシステムソフトウェアーV1.1⑤グラフィックツール/TOWNS PAINT V1.1⑥OS/MS-DOSエミュレータV1.1

①〜⑥計　標準価格￥538,000
ご奉仕大特価￥448,000

クレジットでも大特価

Aセット ※分割の一例です。ボーナス併用、または一括払もございます。

	初回	毎月		
1.	初回￥15,698	毎月￥12,400×35回	税込支払合計	￥449,698
2.	〃 ￥19,732	〃 ￥17,700×23回	〃	￥426,832
3.	〃 ￥34,366	〃 ￥33,600×11回	〃	￥403,966

Bセット ※クレジット金額には消費税が含まれております。

	初回	毎月		
4.	初回￥18,814	毎月￥13,700×35回	税込支払合計	￥498,314
5.	〃 ￥22,176	〃 ￥19,600×23回	〃	￥472,976
6.	〃 ￥38,438	〃 ￥37,200×11回	〃	￥447,638

●シャープMZ-1×30 モデムホン

標準価格￥98,000⇨
特価￥39,800
〈300/1200BPS全2重通信対応モデム内蔵〉
〈音声入出力端子付〉
〈ダイヤルパルス/プッシュボタン対応〉
〈プッシュボタン音解析機能〉
〈シャープ手順、CCITT、V25bis通信手順サポート〉

●東芝BOOK・Computer J3100SS

標準価格￥198,000⇨ 大特価！

アイビット推奨ディスプレイ

●富士通ゼネラルDM405
(14型)
(2000アナログ21/8ピン)
定価￥67,800⇨
特価￥36,000

DM405対応パソコン機種：MSX2。X1シリーズ。MZ700/1500/2000/2200シリーズ。FM77AV/7/8シリーズ。（ケーブルは各専用のものを使用）

●シャープCZ-830D・BK
(14型)
2モードオートスキャン方式
（アナログ/デジタル）
定価￥98,000⇨
特価￥54,800

CZ-830D対応パソコン機種：CZ880C/881C。X1/TURBOシリーズ。ケーブルは本体付属を使用。PC88VA/VA2/VA3/MK2SR/TR/FR/MR。PC9801U/UV/UX/VM/VX/LV各シリーズ。アナログ25ピン↔25ピンケーブルを使用（デジタルは各専用ケーブルで）。MZ700/1500/2000/2200/2500各シリーズ（推奨品シャープ8D8K）。

●シャープCZ-611D-GY
(15型アナログTV/3モードオートスキャン)
￥145,000→￥89,800

CZ-611D対応パソコン機種：※X1シリーズ/※X1 turboシリーズ/X1 yurboZシリーズ/X68000シリーズ/PC8801シリーズ/PC-9801シリーズ/PC-286シリーズ
（※は接続ケーブルAN1506が必要です）

●三菱XC-1498C
(14型アナログ/ドットピッチ0.28㎜)
定価￥99,800⇨
特価￥54,800

XC-1498C対応パソコン機種：
NEC・PC9801シリーズ。
エプソンPC286/386シリーズ。

本 体／新旧在庫機種(新品)

●シャープ/CZ-601C/CZ-602C/CZ-612C/CZ-652C/CZ-662C/CZ-801C/CZ-802C/CZ-803C/CZ-804C/CZ-820C/CZ-822C/CZ-888C/MZ-2200/MZ-2861/MZ-3500/MZ-5511/MZ-6556
●富士通/FM-NEW7/FM77AV/FM77AV1/FM77AV2/FM77AV20/FM77AV40/FM77D2/FM77L2/TOWNS1/TOWNS2
●東芝/J-3100SL/J-3100SS
●NEC/PC9801CV21/PC9801E/PC9801LV21/PC9801RA2/PC9801RX2/PC9801UV21/PC9801VX4/PC9801XA2

拡張機器他

●シャープCZ-8GR(X1.GRAM)‥￥32,000⇨￥12,000
●シャープCZ-52F(F増設ドライブ代品)…………￥15,000
●シャープCZ-51F(ターボ増設ドライブ代品)…￥15,000
●シャープCZ-8EP(I/Oポート)…￥11,800⇨￥9,000
●シャープCZ-8EB3(I/Oボックス)‥￥33,800⇨￥28,000
●シャープCZ-8BK3…(X1)…￥13,800⇨￥11,700
●シャープCZ-8BK4…(X1)……￥6,800⇨￥5,700
●シャープCZ-8BGR2・(X1)……￥14,800⇨￥4,000
●シャープCZ-8BS1…(X1)…￥23,800⇨￥19,500
●シャープCZ-64H(ハードディスク)〈CZ-602・652内蔵用〉…￥120,000
●シャープCZ-8NJ2(インテリジェントコントローラー)￥23,800⇨大特価
●シャープCZ-8SS2システムスタンド…￥5,500⇨￥2,500
●シャープCZ-81Tチルトスタンド……￥8,500⇨￥1,000
●シャープCZ-8RL1(データーレコーダー)…￥24,800⇨￥16,000
●シャープMZ-1U08(1500,700拡張)‥￥25,000⇨￥12,000
●シャープMZ-1U03(1500,700拡張)‥￥35,000⇨￥15,000
●シャープMZ-1X22モデムユニット…￥21,800⇨￥13,000
●シャープMZ-1R12 RAM………￥35,000⇨￥8,000
●シャープMZ-1E29 (MZ)……￥17,800⇨￥9,800
●シャープMZ-1U09…(2500)…￥9,000⇨￥7,200
●シャープMZ-1M03…(5500)・￥69,000⇨￥35,000
●シャープMZ8BC04‥(2000)‥￥18,000⇨￥8,000
●シャープMZ-8B104‥(2000)・￥45,000⇨￥18,000
●シャープMZ-1R11…(5500)・￥80,000⇨￥40,000
●シャープMZ-1R24…(2500)‥￥22,000⇨￥6,000
●シャープMZ-1R26A‥(2500)・￥13,000⇨￥12,800
●シャープMZ-1R27A‥(2500)・￥13,000⇨￥10,000
●シャープMZ-1R28A・(2500) ￥13,000⇨￥10,000
●シャープMZ-1R29A‥(2500)・￥32,000⇨￥10,000
●シャープMZ-1T02…(2000)‥￥19,800⇨￥8,500
●シャープMZ-1T03…(1500)‥￥12,000⇨￥8,500
●シャープMZ-1X29…………￥13,800⇨￥11,000
●シャープMZ1R35(1MB増設メモリーボード)‥￥55,000⇨￥19,000
●シャープMZ1R36(1R35用増設1MBボード)￥45,000⇨￥15,000
●シャープMZ1E26(ボイスコミュニケーション)…￥24,800⇨￥13,000
●シャープMZ-1R36(MZ2861増設RAM)…￥45,000⇨￥15,000
●シャープMZ-3500キーボード…………￥10,000
●シャープMZ-5500キーボード…………￥10,000
●シャープ2000/2200キーボード …………￥10,000
●シャープSSSC28M(モノクロハンディスキャナー)￥49,800⇨￥10,000
●シャープSS-SC28M(ハンディーコピーキット)￥49,800⇨￥10,000
●シャープ1E35(ADPCMボード)……￥49,800⇨￥13,000
●シャープ1E39(RE232C 2CHボード)…￥39,800⇨￥13,000
●シャープX1、MZ用マウス…………特価￥4,800
●シャープX1用ジョイカード………………￥1,500
●富士通16βキーボード……￥25,000⇨￥20,000

プリンター

●シャープCZ-8PK2(漢字プリンター)・￥134,000⇨￥25,000
●シャープCZ-8PK7(漢字プリンター)・￥122,000⇨￥97,600
●シャープCZ-8PK8(漢字プリンター)￥152,000⇨￥121,500
●シャープCZ-8PK9(漢字プリンター)…￥89,800⇨￥71,800
●シャープCZ-8P1(801用プロッタプリンタ)……￥3,500
●シャープCZ-8PC2…………￥69,800⇨￥46,800
●シャープCZ-8PC3…………￥65,800⇨￥52,000
●シャープCZ-8PC4(黒・グレー)・￥99,800⇨大特価
●シャープCZ-8PD3(X1用)……￥59,800⇨￥16,000
●シャープMZ-1P27………￥268,000⇨￥214,400
●シャープMZ-1P28………￥148,000⇨￥118,400
●シャープMZ-1P29………￥168,000⇨￥134,400
●シャープ6P-11(カットシードヒート)…￥95,000⇨￥35,000
●富士通FMPR-201…………￥79,800⇨￥45,000
●富士通FMPR-351………￥250,000⇨￥125,100
●富士通FMPR-353………￥198,000⇨￥115,000
●富士通MB-27409…………￥98,000⇨￥45,000
●富士通MB-27413…………￥90,000⇨￥25,000
●富士通FMPR-201R1 (二水ROM) ￥23,000⇨￥11,000
●富士通MB27407(熱転写プリンタ)…￥79,800⇨￥33,000
●NEC-NM9700(漢字プリンタ)‥￥163,000⇨￥88,000

ディスプレー（カラー）

●富士通FMTV-211(200)…￥185,000⇨￥89,000
●富士通FMTV-152(200)…￥109,000⇨￥58,000
●NEC PC-KD854(400)……￥89,800⇨￥58,000

ディスプレー（モノカラー）

●シャープCZ-1D10(400)…￥41,800⇨￥25,000
●NEC PC-8050(200)………￥29,800⇨￥24,000

フロッピーディスク

●シャープCZ-503F…………￥49,800⇨￥34,000
●シャープ CZ-503F(インターフェースカードなし)…￥30,000
●シャープCZ-502F…………￥99,800⇨￥75,000
●シャープCZ-300F(CZ-3PCM付)…………￥13,000

ソフト

●ユーカラK2+………(2500)・￥28,000⇨￥23,000
●春望クリエイティブII・(2500)・￥34,800⇨￥29,000
●ビジレス……………(2500)・￥48,000⇨￥42,000
●Hu-CAL日本語……(2500)・￥45,000⇨￥30,000
●ぷりんとしょっぷ……(2500)…￥9,800⇨￥5,000
●G.EDIT2500…………(2500)…￥8,000⇨￥7,000
●FILE UTILITY UT-25F・(2500)…￥6,800⇨￥6,000
●パーソナルCP/M6Z001(2500)……………11月入荷
●V2BASIC6Z010……(2500)‥￥10,000⇨￥8,500
●FORTRAN(1P1213)‥(2500)・￥13,800⇨￥11,700
●C MZ2500 1P1214‥(2500)・￥13,800⇨￥11,700
●COBOL 1P1215…(2500)・￥13,800⇨￥11,700
●C CZ116LF(X1)……………￥13,800⇨￥11,700
●COBOL CZ118LF‥(X1)…￥13,800⇨￥11,700
●ランゲージマスターCZ128SF…￥9,800⇨￥8,500
●シャープCZ-130F(ターボCP/M)…￥14,800⇨￥12,500
●シャープCZ-133SF(モデムターミナル)‥￥25,000⇨￥6,500
●シャープX1・3インチCP/M…￥16,800⇨￥5,000
●富士通B273D030(サウンドエディタ)……￥9,800⇨￥3,000
●富士通B273D040(ミュージックエディタ)…￥9,800⇨￥3,800
●富士通B273D050(グラフィックエディタ)…￥9,800⇨￥3,000
●HUMAN68K CZ-244SS……￥9,800⇨￥8,500

X68000関係ソフト

●マイクロソフトウェアージャパン「C&プロフェッショナルパッケージ」…………￥58,000⇨￥49,800
●シャープOS-9/X68000……￥29,800⇨￥25,300
●シャープCZ-211LS…………￥39,800⇨￥33,800
●シャープCZ-6BE1…………￥35,000⇨￥29,000
●シャープCZ-6BE1A…………￥38,000⇨￥32,000

■シャープポケコン全商品販売中。カタログ、特価表ご請求ください(〒72)。

☎0426-45-3001〜3
FAX.0426-44-6002
●営業時間/10:00〜19:00●電話受付/20:00迄可●定休日/日曜日(祭日営業)
SHARP SUPER XEX SHOP
アイビット電子株式会社 〒192 東京都八王子市北野町560-5

上記の広告商品はすべて店頭販売もしております。

★送料はご注文の際にお問い合わせ下さい。
★掲載の商品は、すべて新品、保証書付きです。
★掲載の商品は充分用意してありますが、ご注文の際は、在庫の確認の上、現金書留または、銀行振込でお申し込み下さい。全商品クレジットでも扱っております。
★お申し込みの際は必ず電話番号を明記して下さい。
★商品、品切れの節はご容赦下さい。

富士銀行八王子支店 (普)1752505

●本誌発売時には、上記価格よりさらにお求めやすい価格に変更されている場合があります。●上記商品価格には消費税は含まれておりません。全ての商品に対し、別途3%の消費税金がかかりますのでご了承ください。

パソコン・AV専門 O.A.ランド

- ●お近くの方は、お立寄り下さい。専門係員がアドバイスいたします。
- ●ビジネスソフト、ゲームソフトのことならおまかせ下さい!!

セール期間
◀ '89 10・16 ➡ 11・15

涼しいときには、ハイ・パピプペポンと オータムセール 実施中!!

安心と信頼のOAランド・優良パソコン販売店、
アフターサービス万全のサポート体制。

NEW ランド特選 SHARP X68000EXPERT・EXPERT HDセット

X68000EXPERT HDセット
40MB HDD内蔵 2MB RAM

- ●CZ-612C ……………… 定価¥466,000
- ●CZ-612D ……………… 定価¥119,800
- ●MD-2HD 20枚サービス

クレジット例：12回…月々¥39,000、24回…月々¥20,400

他店には負けません!! 合計定価¥585,800

現金大特価!!

ゲームソフト 5ゲームプレゼント

組合せは自由ダョ!!

X68000EXPERTセット
2MB RAM内蔵

- ●CZ-602C ……………… 定価¥356,000
- ●CZ-612D ……………… 定価¥119,800
- ●MD-2HD 20枚サービス

クレジット例：12回…月々¥31,500、24回…月々¥16,500

OAランドで買わなきゃ損をする! 合計定価¥475,800

現金大特価!! 大推選!!

ゲームソフト 5ゲームプレゼント

NEW X-1ターボZⅢセット
CRTクリーナー キーボードカバープレゼント

Ⓐセット
- ●CZ-888CBK … 定価¥169,800
- ●CZ-880DBK … 定価¥109,800
- ●CZ-6ST1B … 定価¥ 5,800 (チルトスタンド)
- ●MD-2HD 20枚サービス

合計定価¥275,400

現金価格 特価中TEL下さい

安すぎてゴメンなさい!

Ⓑセット
- ●CZ-888CBK … 定価¥169,800
- ●CZ-830DBK … 定価¥ 98,000
- ●CZ-6ST-1B … 定価¥ 5,800 (チルトスタンド)
- ●MD-2HD 20枚サービス

合計価格¥273,600

合計価格 特価中TEL下さい

NEW SHARP X68000 PRO・PRO HDセット

ゲームソフト 5ゲームプレゼント

X68000PROセット
- ●CZ-652C ……… 定価¥298,000
- ●CZ-612D ……… 定価¥119,800
- ●MD-2HD 20枚サービス

クレジット例：12回…月々¥27,800、24回…月々¥14,500

合計定価¥417,800

現金特価!! TEL下さい。

X68000PRO-HDセット
- ●CZ-662C ……… 定価¥408,000
- ●CZ-612D ……… 定価¥119,800
- ●MD-2HD 20枚サービス

クレジット例：12回…月々¥34,900、24回…月々¥18,300

合計定価¥527,800

現金特価!! TEL下さい。

X68000 お買徳!! X-1TWIN

展示新同品
- ●CZ-611C(GY)
- ●CZ-611D(GY)

2セット限り

現金特価 ¥328,000

新同品
- ●CZ-830C 定価¥99,800

PCエンジン内蔵

現金特価 ¥38,000

周辺機器コーナー

X1用

- ●CZ-8BV2 … 定価¥ 39,800 ▶ 特価¥ 31,000
- ●CZ-8BR1 … 定価¥ 29,800 ▶ 特価¥ 23,000
- ●CZ-8DT2 … 定価¥ 44,800 ▶ 特価¥ 35,000
- ●CZ-8BS1 … 定価¥ 23,800 ▶ TEL下さい
- ●CZ-8TM2 … 定価¥ 49,800 ▶ 特価¥ 38,000
- ●CZ-8EB3 … 定価¥ 33,800 ▶ 特価¥ 27,000

X68000用

- ●CZ-6PU1A … 定価¥ 38,000 ▶ 特価¥ 30,000
- ●CZ-6BM1 … 定価¥ 26,800 ▶ 特価¥ 21,000
- ●CZ-6BE1 … 定価¥ 88,000 ▶ 特価¥ 69,800
- ●CZ-6VT1 … 定価¥ 69,800 ▶ TEL下さい
- ●CZ-8NS1 … 定価¥188,000 ▶ 特価¥149,000
- ●CZ-6BC1 … 定価¥ 79,800 ▶ 特価¥ 63,000

プリンターセットコーナー

①CZ-6PU1(カラービデオプリンター) 定価¥198,000 ▶ 特価¥152,000
②CZ-8PC3(カラープリンター) …… 定価¥ 65,800 ▶ 特価¥ 53,000
③CZ-8PK8(ドットプリンター) …… 定価¥152,000 ▶ 特価¥115,000
④CZ-8PK7(ドットプリンター) …… 定価¥122,000 ▶ 特価¥ 93,000
⑤PC-PR201TH(カラープリンター) 定価¥145,000 ▶ 特価¥103,000
⑥PC-PR201G(ドットプリンター) … 定価¥158,000 ▶ 特価¥ 99,000

その他、周返機器・プリンター ソフトウェアー 20%〜25% OFF!!

X68000用ソフトウェアー・コーナー

①CZ-212BS(BUSINESS) …… 定価¥ 68,000 ▶ 特価¥ 53,000
②CZ-220BS(DATA) …… 定価¥ 58,000 ▶ 特価¥ 45,000
③CZ-215MS(Sampling) …… 定価¥ 17,800 ▶ 特価¥ 13,800
④CZ-221HS(NEW Print Shop) … 定価¥ 10,800 ▶ 特価¥ 15,500
⑤CZ-227BS(TOP財務会計) … 定価¥200,000 ▶ 特価¥158,000
⑥CZ-226BS(CARD) …… 定価¥229,800 ▶ 特価¥ 23,000
⑦CZ-223CS(Communication) … 定価¥ 19,800 ▶ 特価¥115,500
⑧CZ-213MS(MUSIC) …… 定価¥ 18,800 ▶ 特価¥ 14,800
⑨CZ-211LS(C compiler) …… 定価¥ 39,800 ▶ 特価¥ 31,000
⑩C-TRACE(キャスト) …… 定価¥ 68,000 ▶ 特価¥ 52,000
⑪EW(イースト) …… 定価¥ 38,000 ▶ 特価¥ 29,000

■ハードディスク ■特価品もありますので TEL下さい。

- ●アイテック IT-MJ4(I/F付) …… 特価¥98,000
- ●アイテック IT-MJ4 C(I/F付) …… 特価¥109,000
- ●ウィンテック HD-404HS(I/F付) … 特価¥108,000
- ●コンピュータ CRC-MH4(I/F付) …… 特価¥70,000
- ●スナイパー SR-340II(I/F付) …… 特価¥78,000
- ●アイテック ITH-320S(I/F付) …… 特価¥79,800
- ●ウィンテック HD-202(I/F付) …… 特価¥58,000
- ●スナイパー SR-520(I/F付) …… 特価¥55,000
- ●コンピュータ CRC-HD2A(I/F付) … 特価¥62,000
- ●ロジテック LHD-32NR(I/F付) …… 特価¥80,000

今月の特価品 各一台限り その他、いろいろありますのでTEL下さい!!

■A紙品(美品・POP品) ■B級品(キズ少々) ■C級品(キズ有り)

		A級品	B級品	C級品
X68000シリーズ	●CZ-611C	¥250,000より	¥245,000	¥238,000
	●CZ-652C	¥219,000より	¥212,000	¥203,000
	●CZ-611D	¥ 90,000	¥ 86,000	¥ 80,000
	●CZ-603	¥ 58,000	¥ 55,000	¥ ―
X-1シリーズ	●CZ-888C	¥ 99,800より	¥ 90,000	―
	●CZ-822C	¥ 24,000より	¥ 20,000	―
	●CZ-880D	¥ 75,000	¥ 71,000	―
	●CZ-830C	¥ 37,000	¥ 33,000	―
X-1プリンター	●CZ-8PC3	¥ 48,000	¥ 45,000	¥ 42,000
	●CZ-7PK7	¥ 83,000	―	―
	●CZ-8PK8	¥109,000	¥105,000	―
	●CZ-6PV1	¥138,000	¥134,000	¥125,000

その他、いろいろありますので、TELください。

中古パソコン(価格・在庫は変動します。予約は5日以内といたします。)

機種	価格	機種	価格
PC-9801VX21	¥220,000より	PC-8801mkII 30	¥ 35,000より
PC-9801VX2	¥195,000より	PC-8801mkII SR	¥ 73,000より
PC-9801VM2	¥158,000より	PC-8801mkII FR30	¥ 68,000より
PC-9801VF2	¥ 98,000より	PC-8801mkII MR	¥ 88,000より
PC-9801M2	¥138,000より	PC-88VA	¥148,000より
PC-9801F2	¥ 78,000より	PC-8801mkII FH30	¥ 85,000より
PC-9801UV21	¥138,000より	PC-8801FA	¥108,000より
PC-98LTM1(640KB)	¥ 89,000より	X-1Gモデル30	¥ 25,000より
PC-286モデル0	¥168,000より	X-1ターボII	¥ 68,000より
		FM-77D2	¥ 28,000より
PC-286V-STD	¥202,000より	FM-77AV2	¥ 42,000より
X-68000	¥188,000より	FM-77AV20	¥ 52,000より

通信販売のご案内
全国通販

■銀行振込で申し込みの方は商品名及びお客様の住所・氏名・電話番号をお知らせ下さい。
〔振込先〕第一勧業銀行 渋谷支店
普通No.1163457 ㈱オーエーランド

■現金書留で送金されるお客様は電話番号と商品名、数量を明記して同封して下さい。■クレジットでご購入を希望される方は申し込み用紙をお送り致しますのでご記入の上返送して下さい。20才以上の方は、原則として保証人不要です。クレジットは1〜60回払で月々5,000円よりご自由に設定できます。

品川、目黒方面 / 首都高速3号線 / O.A.ランド / 渋谷駅 / 井の頭線渋谷駅 / 道玄坂 / ヤマハ / J&P / 109 / 神泉駅 / 明治通り / 山手線 / 西武百貨店 / 東急百貨店

- ●下取・買取は電話で見積りしております。責任を持って下取りさせて頂きます。
- ●ご注文、お問合せは…毎日午前10時から午後7時まで
- ●商品のお届けは…入金確認後、即日発送致します。

㈱オーエー・ランド

〒150東京都渋谷区円山町20-4 第5日新ビル1F
☎(03)770-8855 FAX(03)770-7080

関東エリアの送料は、1個につき¥1,000です。

★全商品保証書付。専門のアドバイザーが、お客様のニーズに対応します。
★初期不良・輸送トラブル等に迅速に対応し、即交換させていただきます。

■表示価格は、税別表示です。詳しくは、お電話にて、お問い合せ下さい。掲載の価格は、9月末現在です。

JPDMA 安心と信頼のシステムで新時代を切り開く

Let's CALL 高額下取りセール実施中!! 今すぐお電話下さい。 03(797)1221

X68000

EXPARTシリーズ・PROシリーズ新登場!!

・オリジナルOS「Human68k ver. 2.0」を搭載
・40MBハードディスクドライブを内蔵
・メインメモリ2MB標準装備(EXPERTシリーズ)
・拡張I/Oスロット4スロット内蔵(PROシリーズ)

☆注文No.A-1121

SHARP CZ-602C	¥356,000
SHARP CZ-602D	¥ 99,800
標準価格合計	¥455,800
現金特別価格	~~¥455,800~~

大特価にて提供中

☆注文No.A-1122

SHARP CZ-612C	¥466,000
SHARP CZ-602D	¥ 99,800
標準価格合計	¥565,800
現金特別価格	~~¥565,800~~

大特価にて提供中

☆注文No.A-1123

SHARP CZ-652C	¥298,000
SHARP CZ-602D	¥ 99,800
標準価格合計	¥397,800
現金特別価格	~~¥397,800~~

大特価にて提供中

☆注文No.A-1124

SHARP CZ-662C	¥408,000
SHARP CZ-602D	¥ 99,800
標準価格合計	¥507,800
現金特別価格	~~¥507,800~~

大特価にて提供中

当社はX68000 PRO SHOPです。

●どこよりもお得な高額下取り実施中!! セットの組合わせは自由自在、ぜひご相談下さい。

X1 turbo Z III

画像取り込み、ビデオ編集、ステレオFM音源、多才な機能でひろがるアートワーク。

☆注文No.A-1125

SHARP CZ-888C-BK	¥169,800
SHARP CZ-860D-BK	¥ 92,200
標準価格合計	¥262,000
現金特別価格	~~¥262,000~~

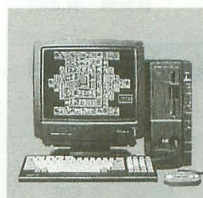
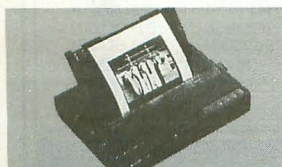
大特価にて提供中

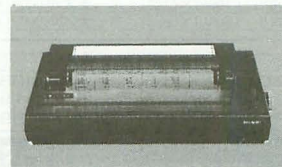
X1 twin

HEシステム(PC Engine)搭載で楽しさ2倍

☆注文No.A-1126

SHARP CZ-830C-BK	¥ 99,800
SHARP CZ-830D-BK	¥ 90,600
標準価格合計	¥190,400
現金特別価格	~~¥190,400~~

大特価にて提供中

●どこよりもお得な高額下取り実施中!! セットの組合わせは自由自在、ぜひご相談下さい。

☆注文No.B-1123
"24ドット熱転写カラー漢字プリンタ"
SHARP CZ-8PC3 ¥65,800
現金特別価格 ~~¥65,800~~
大特価にて提供中
■お支払例
①¥10,000×6回〔ボーナス〕無し
②¥ 3,200×20回〔ボーナス〕無し

☆注文No.B-1125
"48ドット熱転写カラー漢字プリンタ"
SHARP CZ-8PC4 ¥99,800
現金特別価格 ~~¥99,800~~
大特価にて提供中
■お支払例
①¥9,500×10回〔ボーナス〕無し
②¥3,000×36回〔ボーナス〕無し

☆注文No.B-1147
"24ピン136桁漢字プリンタ"
SHARP CZ-8PK8 ¥152,000
現金特別価格 ~~¥152,000~~
大特価にて提供中
■お支払例
①¥ 6,400×24回〔ボーナス〕無し
②¥12,100×12回〔ボーナス〕無し

☆注文No.B-1132
"インテリジェントコントローラ"
SHARP CZ-8NJ2 ¥23,800
現金特別価格 ~~¥23,800~~
大特価にて提供中
■お支払例
①¥3,300×24回〔ボーナス〕無し
②¥6,200×12回〔ボーナス〕無し

中古在庫リスト

SHARP

本体

品名	定価	価格
CZ-812C(X-1 Fmode120)	¥139,800	¥26,000
CZ-822CB(X-1 Gmode130) 新品同様	¥118,000	¥29,800
CZ-850C(X-1 Turbo mode110)	¥168,000	¥22,000
CZ-611C(X68000ACEHD) 新品同様	¥399,800	¥238,000
MZ-2861	¥328,000	¥148,000

ディスプレイ

品名	定価	価格
CU-14G(14" 2000文字カラーディスプレイ)	¥49,800	¥20,000
14M-522C(14" 4000文字デジタルカラーディスプレイ)	¥99,800	¥42,000
CU-14AG1(14" 4000文字アナデジカラーディスプレイ)	¥89,800	¥43,000
CU-14H1(14" 4000文字デジタルカラーディスプレイ)	¥99,800	¥42,000
CU-14BD(14" カラー4050/2000文字)	¥64,800	¥42,000
CU-14CD(14" カラー4050/2000文字) 新品同様	¥84,800	¥52,800
CU-14FD(14" 4000文字アナログカラーディスプレイ) 新品同様	¥74,800	¥51,000
MZ-1D22(14" 4000文字MZ用カラーディスプレイ)	¥108,000	¥45,000

その他

品名	定価	価格
CZ-611D(15" 3モードスキャンカラーディスプレイTV) 新品同様	¥134,000	¥82,000
CZ-8PK7(10" 24ドット漢字プリンタ)	¥122,000	¥52,000
CZ-8PD2(80桁ドットプリンタ)	¥79,800	¥28,000
MZ-1P07(80桁漢字カラーサーマルプリンタ)	¥79,800	¥30,000
CZ-8SS2(システムスタンド) 新品同様	¥5,500	¥4,000
CZ-8NM2(X1用マウス)	¥6,800	¥3,000

SHARP
CZ-611CGY 新品同様
(X68000 ACE HD)
¥399,800→¥238,000
X68000 ACE HDディスプレイセット
(本体+CZ-611DGY) 新品同様
¥533,800→¥320,000

その他各種在庫をとりそろえております。御気軽にお問い合せ下さい。

全商品保証付 中古も6ヶ月の保証期間だから安心です。	クレジットでOK カレッジクレジットも取扱います。
全国無料配送 お買上1万円以上、配達料はいただきません。	日曜配達可 留守の多い方でも安心です。
ショールーム Xシリーズ展示中。	高額買取り 電話1本で即、現金お支払い。
代金引換えシステム 商品到着時の代金支払いでOK。	ボーナス一括払い 商品は即お手元へ、お支払いはボーナス時に。

●電話一本で高額下取り、即商品はお手元へ!
●あなたの不要になったパソコンを電話一本で査定し買取ります。
●掲載の商品以外も取り扱っております。
●ビジネスソフトスクール受講者受付中!
お気軽にお電話下さい。

▼本社注文デスク
03(797)1221
コンピュータバンク

株式会社パシフィックコンピュータバンク 〒150 東京都渋谷区渋谷1-6-8 井上ビル 営業時間/平日AM9:30〜PM9:00 土・休日AM9:30〜PM8:00 年中無休

●クレジット価格に消費税は含まれておりますが、現金特別価格には含まれておりません 別途消費税がかかります

ACCESS

X1エミュレータ

好評発売中

定価¥9,800

X1エミュレータはX68000上でX1シリーズのアプリケーションを実行するためのソフトエミュレータです。X1のアプリケーションを完全にソフトウェアのみでエミュレートしているため、X1上での実行速度と比較して、平均3〜5倍程度おそくなりますが、X68000のマシン上に実現した仮想X1マシンを楽しめます。また、X1とX68000の相互間でファイルを転送するためのユーティリティと専用ケーブルが付属しますので、X1上で作り上げたソフトの資産をX68000上に移行することも簡単にできます。

X1エミュレータの機能

- X1エミュレータはX1に相当する機能をエミュレート。この仮想コンピュータには最大4つのドライブが仮想的に接続。
- X1エミュレータからみたドライブはHuman68kのドライブ上にあるファイルで仮想的に実現。このファイルはX1用の5" 2Dディスクのイメージをファイル転送ユーティリティでまるごと転送したもの。
- X1エミュレータで仮想的に実現したX1は仮想ドライブから起動。このため仮想ドライブ用ファイルには、X1を立ち上げるために必要なHuBASICやCP/Mなどのシステムプログラムが必要。
- X1エミュレータでは、X1の持つVRAMを含むメモリイメージやZ80CPUを仮想的にソフトウェアで実現。

ファイル転送ユーティリティ

ディスク転送

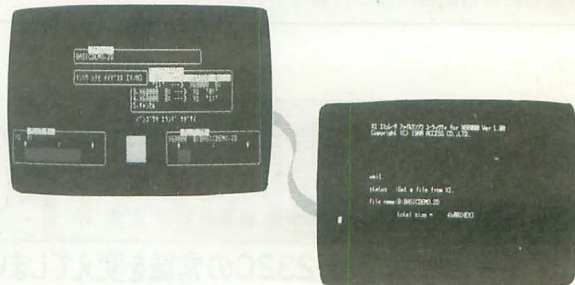
X1ディスク⟷X68000 Human68k(5" 2Dディスクイメージファイル)

- X1エミュレータではHuman68k上のディスクイメージファイルを仮想ドライブとして使用。

ファイル転送

X1 BASIC:CP/M⟷X68000 Human68k

- X1で作ったプログラム&データをX68000上で使用。

※付属の専用ケーブルをX1とX68000に接続してファイルを転送します。

X1エミュレータ Q&A

Q. ファイル転送のために別途RS-232Cケーブルを買わないといけないのですか?

A. 専用のケーブルが付属しますのでその必要はありません。

Q. X1BASICのプログラムをX68000上のX-BASICで使えますか?

A. 通常のセーブではコードが違うので使用できませんが、アスキーセーブしたファイルであればX-BASIC上でそのままロード可能です。

Q. TurboBASICで作成した住所録などの漢字を含んだデータがあるのですがX68000上にファイル転送できますか?

A. X1TurboもX68000も漢字はシフトJISコードなのでファイルの転送は可能です。ただし、漢字ROMを必要とするものはサポートしていません。

Q. Turbo用のソフトは動きますか?

A. X1用のみでTurbo専用のソフトは動きません。

Q. ゲームは動きますか?

A. 純粋にBASICでかかれたものは動きますが、プロテクトがかかったものや直接ハードをアクセスするような市販のゲームは動きません。

*タイミング等ハードウェアに依存するようなソフトは、原理上実行できない、もしくは正常に動作しない場合がありますのでご注意ください。
*一部サポートしていない機能があります。

X1エミュレータ通信販売 購入希望として住所、氏名、電話番号をお知らせください。注文書をお送り致します。

発売中　X68000用　**CONCERTO-X68K**　MS-DOSエミュレータ　定価¥99,800

代理店募集

アクセスではこれらの製品の発売にあたり代理店を募集しております。詳しくはお問い合せください。

*この商品価格には消費税は含まれておりません。
*MS-DOSはマイクロソフト社, CP/Mはデジタルリサーチ社の商標です。
文中のソフトウェアは各社の商標です。
*製品の仕様、名称は予告なく変更する場合もございますのであらかじめご了承ください。

有限会社アクセス　〒101 東京都千代田区神田神保町1-64 神保町協和ビル7F
☎03(233)0200(代) FAX.03(291)7019

資料請求券 oh!X 11月号

T4910217911562 雑誌02179-11